UNIVERSIDAD DE GUAYAQUIL

Facultad de Ciencias Matemáticas y Físicas

Carrera de Ingeniería en Sistemas Computacionales

"Desarrollo del Módulo de Ventas de un sistema ERP"

MANUAL TÉCNICO Y USUARIO

Previo a la Obtención del Título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Autores:

GABRIEL GIANCARLO GALLO MACIAS

DANNY GEOVANNY REYES POVEDA

FRANCISCO XAVIER TRIVIÑO ARMENDÁRIZ

GUAYAQUIL – ECUADOR

Año: 2006

INDICE GENERAL

MANUAL TÉCNICO

Clase Documento Comercial 2

Clase Forma de Pago 5

Clase Comisión 6

Clase Vendedor Bono 7

Factura DAO 10

Nota de crédito DAO 23

Vendedor DAO 33

Comisión DAO 37

MANUAL DE USUARIO

Consulta de Facturas 61

Creación de Facturas 65

Consulta de Notas de Venta 69

Creación de Notas de ventas 70

Consulta de Proformas 73

Creación de Proformas 74

Consulta de Nota de Crédito 77

Creación de Notas de Crédito 78

Creación de Nuevo Vendedor 80

Crear Nuevo Tipo de Vendedor 80

Crear un Nuevo Tipo de Comisión 81

Comisiones 81

Reportes 83

Ventas por Fecha 83

Ventas por Vendedor 85

Devoluciones por fecha 86

Ventas de Artículos 86

Libro de Ventas 87

Ventas por Clientes 88

Artículos más vendidos 89

General Vendedores 90

Reporte de Comisiones generales 90

Lista de Precios 91


MANUAL TÉCNICO

El desarrollo de este programa se lo realizó en lenguaje JAVA por lo que a continuación detallamos las clases y procesos más importantes de nuestro sistema:

CLASE DOCUMENTO COMERCIAL

```
package com.ug.cisc.erp.ventas.entity;

import java.util.Date;

public class DocumentoComercial {

    long codiSucursal;
    long serie;
    Date fechaRegistro;
    String observacion;
    double iva;
    double subTotal;
    double subTotalParaComision;
    double subTotalPenalizadoParaComision;
    double totalDcto;
    double porcDesc;
    double total;
    String direccionEnvio;

    String estado;

    public DocumentoComercial() {
        super();
    }

    public long getCodiSucursal() {
        return codiSucursal;
    }

    public void setCodiSucursal(long codiSucursal) {
        this.codiSucursal = codiSucursal;
    }

    public long getSerie() {
        return serie;
    }

    public void setSerie(long serie) {
```

```
    this.serie = serie;
}

public Date getFechaRegistro() {
    return fechaRegistro;
}

public void setFechaRegistro(Date fechaRegistro) {
    this.fechaRegistro = fechaRegistro;
}

public String getObservacion() {
    return observacion;
}

public void setObservacion(String observacion) {
    this.observacion = observacion;
}

public double getIva() {
    return iva;
}

public void setIva(double iva) {
    this.iva = iva;
}

public double getSubTotal() {
    return subTotal;
}

public void setSubTotal(double subTotal) {
    this.subTotal = subTotal;
}

public double getSubTotalParaComision() {
    return subTotalParaComision;
}

public void setSubTotalParaComision(double subTotalParaComision) {
    this.subTotalParaComision = subTotalParaComision;
}

public double getSubTotalPenalizadoParaComision() {
```

```
        return subTotalPenalizadoParaComision;
    }

    public void setSubTotalPenalizadoParaComision(double
subTotalPenalizadoParaComision) {
        this.subTotalPenalizadoParaComision =
subTotalPenalizadoParaComision;
    }

    public double getTotalDcto() {
        return totalDcto;
    }

    public void setTotalDcto(double totalDcto) {
        this.totalDcto = totalDcto;
    }

    public double getTotal() {
        return total;
    }

    public void setTotal(double total) {
        this.total = total;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getDireccionEnvio() {
        return direccionEnvio;
    }

    public void setDireccionEnvio(String direccionEnvio) {
        this.direccionEnvio = direccionEnvio;
    }

    public double getPorcDesc() {
        return porcDesc;
    }
```

```
    public void setPorcDesc(double porcDesc) {
        this.porcDesc = porcDesc;
    }
}
```

CLASE FORMA DE PAGO

```
package com.ug.cisc.erp.ventas.entity;

public class FormaPago {
	long codi_forma_pago;
	String descripcion;
	String estado;

	public FormaPago() {
		super();
		// TODO Auto-generated constructor stub
	}

	public long getCodi_forma_pago() {
		return codi_forma_pago;
	}

	public void setCodi_forma_pago(long codi_forma_pago) {
		this.codi_forma_pago = codi_forma_pago;
	}

	public String getDescripcion() {
		return descripcion;
	}

	public void setDescripcion(String descripcion) {
		this.descripcion = descripcion;
	}

	public String getEstado() {
		return estado;
	}

	public void setEstado(String estado) {
		this.estado = estado;
	}

}
```

CLASE COMISION

```
package com.ug.cisc.erp.ventas.entity;
public class Comision {
	long codiComision;
	String tipoComision;
	String descripcion;
	String estado;

	public Comision() {
		super();
		// TODO Auto-generated constructor stub
	}

	public long getCodiComision() {
		return codiComision;
	}

	public void setCodiComision(long codiComision) {
		this.codiComision = codiComision;
	}

	public String getDescripcion() {
		return descripcion;
	}

	public void setDescripcion(String descripcion) {
		this.descripcion = descripcion;
	}

	public String getEstado() {
		return estado;
	}

	public void setEstado(String estado) {
		this.estado = estado;
	}

	public String getTipoComision() {
		return tipoComision;
	}

	public void setTipoComision(String tipoComision) {
		this.tipoComision = tipoComision;
```

```
    }
}
```

CLASE VENDEDOR BONO

```
package com.ug.cisc.erp.ventas.entity;

public class VendedorBono {
    long codiVendedorBono ;
    long codiVendedor ;
    String mesComision ;
    double valorVenta ;
    double valorComision ;
    double valorComisionPagar;
    double valorDevolucion ;
    long  numeroVenta ;
    String observacion ;
    String estado ;

    public String getObservacion() {
        return observacion;
    }

    public void setObservacion(String observacion) {
        this.observacion = observacion;
    }

    public double getValorComisionPagar() {
        return valorComisionPagar;
    }

    public void setValorComisionPagar(double valorComisionPagar) {
        this.valorComisionPagar = valorComisionPagar;
    }

    public long getNumeroVenta() {
        return numeroVenta;
    }

    public void setNumeroVenta(long numeroVenta) {
```

```
        this.numeroVenta = numeroVenta;
}

public double getValorDevolucion() {
        return valorDevolucion;
}

public void setValorDevolucion(double valorDevolucion) {
        this.valorDevolucion = valorDevolucion;
}

public long getCodiVendedor() {
        return codiVendedor;
}

public void setCodiVendedor(long codiVendedor) {
        this.codiVendedor = codiVendedor;
}

public long getCodiVendedorBono() {
        return codiVendedorBono;
}

public void setCodiVendedorBono(long codiVendedorBono) {
        this.codiVendedorBono = codiVendedorBono;
}

public String getEstado() {
        return estado;
}

public void setEstado(String estado) {
        this.estado = estado;
}

public String getMesComision() {
        return mesComision;
}

public void setMesComision(String mesComision) {
        this.mesComision = mesComision;
}

public double getValorComision() {
```

```
		return valorComision;
	}

	public void setValorComision(double valorComision) {
		this.valorComision = valorComision;
	}

	public double getValorVenta() {
		return valorVenta;
	}

	public void setValorVenta(double valorVenta) {
		this.valorVenta = valorVenta;
	}

	public VendedorBono() {
		super();
		// TODO Auto-generated constructor stub
	}

}
```

LOS OBJETOS DE ACCESO A BASE DATOS

FACTURA DAO

```
package com.ug.cisc.erp.ventas.dao;
import com.ug.cisc.erp.ventas.entity.*;

import java.util.ArrayList;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Iterator;

public class FacturaDAO {
   String error;

   public FacturaDAO() {
      super();
      // TODO Auto-generated constructor stub
   }

   public int idFactura() throws SQLException, Exception {
      ResultSet rs = null;
      int idFactura = 0;
      String queryString = ("SELECT MAX(codi_factura) as codi_factura
FROM ven_factura;");
      DBConectionDAO con = new DBConectionDAO();
      PreparedStatement ps = con.prepareStatement(queryString);
      rs = ps.executeQuery(queryString);
      if(rs.next()){
         idFactura = rs.getInt("codi_factura");
      }

      if (rs != null)
         rs.close();
      con.disconnect();

      return idFactura;
   }
```

```
    public void eliminaCart(Iterator contents)throws Exception {
        try{
            while (contents.hasNext()){
                contents.next();
                contents.remove();
            }
        } catch (Exception e) {
            error = e.toString();
            throw new Exception(error);
        }
    }

    public long verificaStock(Iterator contents)
    throws SQLException, Exception {
        DetalleFactura df = new DetalleFactura();
        long codi_arti = 1;
        try {
            while (contents.hasNext()){
                df = (DetalleFactura)contents.next();

                /*Disminuye la cantidad del stock*/
                long cantidad = df.getCantidad();
                //long codi_arti = df.getCodiArti();
                ArticuloDAO artd = new ArticuloDAO();
                Articulo art = artd.buscaArticulos(df.getCodiArti());
                long stock = art.getStock();
                // Validacion del stock del articulo
                if (cantidad > stock){
                    codi_arti = df.getCodiArti();
                }
            }
        } catch (SQLException sqle) {
           error = sqle.toString();//"SQLException: update failed, possible
duplicate entry";
            throw new SQLException(error);
        }
        return codi_arti;
    }

    public long guardar(Factura factura,Iterator contents)
    throws SQLException, Exception {
        String queryString = "";
        DBConectionDAO con = new DBConectionDAO();
```

```
long fac = 0;

try {
   con.setAutoCommit(false);
   PreparedStatement ps;

   queryString = "INSERT  INTO ven_factura (codi_cliente,
codi_proforma, codi_vendedor,  " +
         "codi_suc_cliente, observacion, subtotal,
subtotal_para_comision, iva, total_dcto, total, estado_control, estado, " +
         "fecha_registro, fecha_cobro, fecha_entrega ) " +
         "VALUES(?,?,?,?,?,?,?,?,?,?,?,?, " +
         "date_trunc('second' , LOCALTIMESTAMP), " +
         "date_trunc('second' , LOCALTIMESTAMP) + interval '30 day', "
+
         "date_trunc('second' , LOCALTIMESTAMP) + interval '1 day'); ";

   if (factura.getFormaPago() == 1){ // Contra Factura

      queryString = "INSERT  INTO ven_factura (codi_cliente,
codi_proforma, codi_vendedor,  " +
            "codi_suc_cliente, observacion,
subtotal,subtotal_para_comision, iva, total_dcto, total, estado_control, estado,
" +
            "fecha_pago, fecha_registro, fecha_cobro, fecha_entrega ) " +
            "VALUES(?,?,?,?,?,?,?,?,?,?,?,?, " +
            "date_trunc('second' , LOCALTIMESTAMP), " +
            "date_trunc('second' , LOCALTIMESTAMP), " +
            "date_trunc('second' , LOCALTIMESTAMP) + interval '30 day',
" +
            "date_trunc('second' , LOCALTIMESTAMP) + interval '1 day');
";

      factura.setEstadoControl("P");
   }

   ps = con.prepareStatement(queryString);

   //select to_char(LOCALTIMESTAMP, 'DD/MM/YYYY HH24:MI:SS')

   ps.setLong(1, factura.getCodiCliente());
   ps.setLong(2, factura.getCodiProforma());
   ps.setLong(3, factura.getCodiVendedor());
   ps.setLong(4, factura.getCodiSucuClie());
```

```
//ps.setDate(4, (java.sql.Date)factura.getFechaRegistro());
ps.setString(5, factura.getObservacion());
ps.setDouble(6, factura.getSubTotal());
ps.setDouble(7, factura.getSubTotalParaComision());
ps.setDouble(8, factura.getIva());
ps.setDouble(9, factura.getTotalDcto());
ps.setDouble(10, factura.getTotal());
ps.setString(11, factura.getEstadoControl());
ps.setString(12, factura.getEstado());
ps.execute();

PreparedStatement psfp;
psfp = con.prepareStatement("insert into ven_factura_forma_pago
(codi_factura,codi_forma_pago) values (?,?);");
psfp.setLong(1, idFactura()+1);
psfp.setLong(2, factura.getFormaPago());
psfp.execute();

psfp = con.prepareStatement("UPDATE ven_cliente SET
monto_ultima_compra = ?, fecha_ultima_compra = LOCALTIMESTAMP
where codi_cliente = ? ");
psfp.setDouble(1, factura.getTotal());
psfp.setLong(2, factura.getCodiCliente());
psfp.execute();

PreparedStatement ps1;
PreparedStatement psu;
psu = con.prepareStatement("update inv_articulo_dat set stoc_arti =
stoc_arti - ? where codi_arti = ?;");

DetalleFactura df = new DetalleFactura();
ps1 = con.prepareStatement("INSERT  INTO ven_detalle_factura
(codi_factura," +
        "codi_arti,codi_empr,cantidad, precio, precio_para_comision,
total, total_para_comision) VALUES(?,?,?,?,?,?,?,?);");
while (contents.hasNext()){
    df = (DetalleFactura)contents.next();
    ps1.setLong(1,idFactura()+1);
    ps1.setLong(2,df.getCodiArti());
    ps1.setLong(3,df.getCodiEmpr());
    ps1.setDouble(4,df.getCantidad());
    ps1.setDouble(5,df.getPrecio());
    ps1.setDouble(6,df.getPrecioParaComision());
    ps1.setDouble(7,df.getTotal());
```

```
            ps1.setDouble(8,df.getTotalParaComision());
            ps1.execute();

            /*Disminuye la cantidad del stock*/
            psu.setLong(1,df.getCantidad());
            psu.setLong(2,df.getCodiArti());
            psu.execute();

            contents.remove();
         }

         queryString = "INSERT INTO ven_varios (codi_doc, direccion_envio,
porcentaje_desc, doc)"+
               "VALUES (?,?,?,'FAC');";
         ps = con.prepareStatement(queryString);
         ps.setLong(1,idFactura()+1);
         ps.setString(2, factura.getDireccionEnvio());
         ps.setDouble(3, (factura.getPorcDesc())/100);
         ps.execute();


         con.commit();
         fac = idFactura();

      } catch (SQLException sqle) {
         con.rollback();
         error = sqle.toString();//"SQLException: update failed, possible
duplicate entry";
         throw new SQLException(error);
      } finally{
         con.disconnect();
      }
      return (fac);
   }

   public Factura buscaFactura(long id_factura) throws SQLException,
Exception {
      ResultSet rs = null;
      Factura f= null;
      String nombre = null;
      String apellido = null;

      String queryString = ("select * from ven_factura a, rh_empleado_dat b,
ven_cliente c where a.codi_cliente = c.codi_cliente and a.codi_vendedor =
```

```
idempleado and a.estado = 'A' and b.estado = 'A' and codi_factura = ? order
by codi_factura desc;");
        DBConectionDAO con = new DBConectionDAO();
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setLong(1,id_factura);
        rs = ps.executeQuery();
        if (rs.next()){
            f= new Factura();
            f.setCodiFactura(rs.getLong("codi_factura"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiProforma(rs.getLong("codi_proforma"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setCodiCliente(rs.getLong("codi_suc_cliente"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setIva(rs.getDouble("iva"));
            f.setTotal(rs.getDouble("total"));
            f.setTotalDcto(rs.getDouble("total_dcto"));
            f.setEstado(rs.getString("estado"));
            f.setEstadoControl(rs.getString("estado_control"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            f.setFechaEntrega(rs.getDate("fecha_entrega"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return f;
    }


    public ArrayList listarFactura(int criterio) throws SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        String queryString;
        Factura f = null;
```

```
        queryString = ("select * "+
                "from ven_factura f,rh_empleado_dat b, ven_cliente c "+
                "where f.codi_vendedor = b.idempleado and f.codi_cliente =
c.codi_cliente "+
                "and date_trunc('day',f.fecha_registro) >= date_trunc('day' ,
LOCALTIMESTAMP) + interval '-6 month' "+
                "and f.estado = 'A' "+
                "and b.estado = 'A' "+
                "order by codi_factura desc");

        if (criterio == 1)
            queryString = ("select * from ven_factura a, rh_empleado_dat b,
ven_cliente c where a.codi_cliente = c.codi_cliente and a.codi_vendedor =
idempleado and a.estado = 'A' and b.estado = 'A' order by codi_factura
desc");

        DBConectionDAO con = new DBConectionDAO();
        PreparedStatement ps = con.prepareStatement(queryString);
        rs = ps.executeQuery();
        while (rs.next()){
            f= new Factura();
            f.setCodiFactura(rs.getLong("codi_factura"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiProforma(rs.getLong("codi_proforma"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setCodiCliente(rs.getLong("codi_suc_cliente"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setIva(rs.getDouble("iva"));
            f.setTotal(rs.getDouble("total"));
            f.setTotalDcto(rs.getDouble("total_dcto"));
            f.setEstado(rs.getString("estado"));
            f.setEstadoControl(rs.getString("estado_control"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
        }

        if (rs != null)
```

```
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarFacturaFecha(String fecha1, String fecha2) throws
SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select
a.*,b.nombre,b.apellido,c.nombre1,c.apellido1 from ven_factura a,
rh_empleado_dat b, ven_cliente c where a.codi_cliente = c.codi_cliente and
a.codi_vendedor = idempleado and a.estado = 'A' and b.estado = 'A' and
date_trunc('day',a.fecha_registro) between ? and ? order by codi_factura
desc");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setString(1,fecha1);
        ps.setString(2,fecha2);
        rs = ps.executeQuery();
        while (rs.next()){
            Factura f= new Factura();
            f.setCodiFactura(rs.getLong("codi_factura"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiProforma(rs.getLong("codi_proforma"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setCodiCliente(rs.getLong("codi_suc_cliente"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setIva(rs.getDouble("iva"));
            f.setTotal(rs.getDouble("total"));
            f.setEstado(rs.getString("estado"));
            f.setEstadoControl(rs.getString("estado_control"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
        }
```

```
        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarFacturaMonto(double montoi, double montof) throws
SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_factura a, rh_empleado_dat b,
ven_cliente c where a.codi_cliente = c.codi_cliente and a.codi_vendedor =
idempleado and a.estado = 'A' and b.estado = 'A' and total between ? and ?
order by total;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setDouble(1,montoi);
        ps.setDouble(2,montof);
        rs = ps.executeQuery();
        while (rs.next()){
            Factura f= new Factura();
            f.setCodiFactura(rs.getLong("codi_factura"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiProforma(rs.getLong("codi_proforma"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setCodiCliente(rs.getLong("codi_suc_cliente"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setIva(rs.getDouble("iva"));
            f.setTotal(rs.getDouble("total"));

            f.setEstado(rs.getString("estado"));
            f.setEstadoControl(rs.getString("estado_control"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
```

```
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarFacturaVendedor(long codi_vendedor) throws
SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_factura a, rh_empleado_dat b,
ven_cliente c where a.codi_vendedor = idempleado and a.codi_vendedor = ?
and a.codi_cliente = c.codi_cliente and a.estado = 'A' and b.estado = 'A' order
by codi_factura desc;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setDouble(1,codi_vendedor);
        rs = ps.executeQuery();
        while (rs.next()){
            Factura f= new Factura();
            f.setCodiFactura(rs.getLong("codi_factura"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiProforma(rs.getLong("codi_proforma"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setCodiCliente(rs.getLong("codi_suc_cliente"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setIva(rs.getDouble("iva"));
            f.setTotal(rs.getDouble("total"));

            f.setEstado(rs.getString("estado"));
            f.setEstadoControl(rs.getString("estado_control"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
```

```
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarFacturaCliente(String criterio) throws SQLException,
Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_factura a, ven_cliente b,
rh_empleado_dat c where a.codi_cliente = b.codi_cliente and c.idempleado =
a.codi_vendedor and upper(b.apellido1) like upper(?) and a.estado = 'A' and
b.estado = 'A' order by codi_factura desc;");
        PreparedStatement ps = con.prepareStatement(queryString);
        criterio+="%";
        ps.setString(1,criterio);
        rs = ps.executeQuery();
        while (rs.next()){
            Factura f= new Factura();
            f.setCodiFactura(rs.getLong("codi_factura"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiProforma(rs.getLong("codi_proforma"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setCodiCliente(rs.getLong("codi_suc_cliente"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setIva(rs.getDouble("iva"));
            f.setTotal(rs.getDouble("total"));
            f.setEstado(rs.getString("estado"));
            f.setEstadoControl(rs.getString("estado_control"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
```

```
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public void actualizaCabecera(Factura fac) throws SQLException,
Exception {
        DBConectionDAO con = new DBConectionDAO();
        try  {
            String queryString = ("update ven_factura " +
                    "set subtotal_penalizado_para_comision = ? "+
                    "where codi_factura = ? ");


            PreparedStatement ps = con.prepareStatement(queryString);

            ps.setDouble(1,fac.getSubTotalPenalizadoParaComision());
            ps.setLong(2,fac.getCodiFactura());

            int i = ps.executeUpdate();

            con.commit();

        } catch (SQLException sqle) {
            con.rollback();
            error = "SQLException: No se ha podido ejecutar esta petición.
Contacte con el webmaster";
            throw new SQLException(error);
        } catch (Exception e) {
            con.rollback();
            error = "Ha ocurrido una excepcion consultando la base de datos.
Contactese con el Webmaster";
            throw new Exception(error);
        } finally {
            con.disconnect();
        }

    }

    public ArrayList buscaDetalle(long id_factura) throws SQLException,
Exception {
```

```
        ResultSet rs = null;
        DetalleFactura df;
        ArrayList lista = new ArrayList();
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_detalle_factura where
codi_factura = ?;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setLong(1,id_factura);
        rs = ps.executeQuery();
        while (rs.next()){
            df = new DetalleFactura();
            df.setCodiFactura(rs.getLong("codi_factura"));
            df.setCodiDetalleFactura(rs.getLong("codi_detalle_factura"));
            df.setCodiArti(rs.getLong("codi_arti"));
            df.setCantidad(rs.getLong("cantidad"));
            df.setPrecio(rs.getDouble("precio"));
            df.setPrecioParaComision(rs.getDouble("precio_para_comision"));
            df.setTotal(rs.getDouble("total"));
            df.setTotalParaComision(rs.getDouble("total_para_comision"));
            lista.add(df);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }
}
```

NOTA DE CRÉDITO DAO

```
package com.ug.cisc.erp.ventas.dao;
import com.ug.cisc.erp.ventas.entity.*;

import java.util.ArrayList;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Iterator;

public class NotaCreditoDAO {
   String error;

   public NotaCreditoDAO() {
      super();
   }

   public int idNota_Credito() throws SQLException, Exception {
      ResultSet rs = null;
      int id_Nota_Credito = 0;
      DBConectionDAO con = new DBConectionDAO();

      String queryString = ("SELECT MAX(codi_nota_Credito) as
codi_nota_Credito FROM ven_nota_Credito;");
      PreparedStatement ps = con.prepareStatement(queryString);
      rs = ps.executeQuery(queryString);
      if(rs.next()){
         id_Nota_Credito = rs.getInt("codi_nota_Credito");
      }

      if (rs != null)
         rs.close();
      con.disconnect();

      return id_Nota_Credito;
   }

   public long guardar(NotaCredito notaCredito,Iterator contents)
   throws SQLException, Exception {
      long nc = 0;
      DBConectionDAO con = new DBConectionDAO();
      try {
```

```
con.setAutoCommit(false);
PreparedStatement ps;

ps = con.prepareStatement("INSERT  INTO ven_nota_Credito
(codi_cliente, codi_vendedor,  " +
        "codi_suc_cliente, observacion, subtotal, iva, total_dcto, total,
estado," +
        "fecha_registro, fecha_vence, subtotal_para_comision) " +
        "VALUES(?,?,?,?,?,?,?,?,?, " +
        "date_trunc('second' , LOCALTIMESTAMP), " +
        "date_trunc('second' , LOCALTIMESTAMP) + interval '30 day',?);
");

ps.setLong(1, notaCredito.getCodiCliente());
ps.setLong(2, notaCredito.getCodiVendedor());
ps.setLong(3, notaCredito.getCodiSucuClie());

ps.setString(4, notaCredito.getObservacion());
ps.setDouble(5, notaCredito.getSubTotal());
ps.setDouble(6, notaCredito.getIva());
ps.setDouble(7, notaCredito.getTotalDcto());
ps.setDouble(8, notaCredito.getTotal());
ps.setString(9, notaCredito.getEstado());
ps.setDouble(10, notaCredito.getSubTotalParaComision());


ps.execute();


PreparedStatement ps1;
PreparedStatement psu;
psu = con.prepareStatement("update inv_articulo_dat set stoc_arti =
stoc_arti + ? where codi_arti = ?;");


DetalleNotaCredito dnv = new DetalleNotaCredito();
ps1 = con.prepareStatement("INSERT  INTO
ven_detalle_nota_Credito (codi_nota_Credito," +
        "codi_arti,codi_empr,cantidad, precio, precio_para_comision,
total, total_para_comision) VALUES(?,?,?,?,?,?,?,?);");
while (contents.hasNext()){
   dnv = (DetalleNotaCredito)contents.next();
   ps1.setLong(1,idNota_Credito()+1);
   ps1.setLong(2,dnv.getCodiArti());
```

```
            ps1.setLong(3,dnv.getCodiEmpr());
            ps1.setDouble(4,dnv.getCantidad());
            ps1.setDouble(5,dnv.getPrecio());
            ps1.setDouble(6,dnv.getPrecioParaComision());
            ps1.setDouble(7,dnv.getTotal());
            ps1.setDouble(8,dnv.getTotalParaComision());
            ps1.execute();

            /*Aumenta la cantidad del stock*/
            psu.setLong(1,dnv.getCantidad());
            psu.setLong(2,dnv.getCodiArti());
            psu.execute();

            contents.remove();
          }

          String queryString = "INSERT INTO ven_varios (codi_doc,
direccion_envio, porcentaje_desc, doc)"+
               "VALUES (?,?,?,'NC');";
          ps = con.prepareStatement(queryString);
          ps.setLong(1,idNota_Credito()+1);
          ps.setString(2, notaCredito.getDireccionEnvio());
          ps.setDouble(3, (notaCredito.getPorcDesc())/100);
          ps.execute();

          con.commit();
          nc = idNota_Credito();

       } catch (SQLException sqle) {
          con.rollback();
          error = sqle.toString();//"SQLException: update failed, possible
duplicate entry";
          throw new SQLException(error);
       } finally {
          con.disconnect();
       }
       return (nc);
    }

    public void eliminaCart(Iterator contents)throws Exception {
       //DetalleFactura df = null;

       try{
          while (contents.hasNext()){
```

```
            contents.next();
            contents.remove();
          }
        } catch (Exception e) {
          error = e.toString();
          throw new Exception(error);
        }
      }

      public NotaCredito buscaNotaCredito(long id_notaCredito) throws
SQLException, Exception {
        ResultSet rs = null;
        NotaCredito nv= null;
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_nota_Credito a,
rh_empleado_dat b, ven_cliente c where a.codi_cliente = c.codi_cliente and
a.codi_vendedor = idempleado and a.estado = 'A' and b.estado = 'A' and
codi_nota_Credito = ? order by codi_nota_Credito desc;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setLong(1,id_notaCredito);
        rs = ps.executeQuery();
        if (rs.next()){
          nv = new NotaCredito();
          nv.setCodiNotaCredito(rs.getLong("codi_nota_Credito"));
          nv.setCodiTipoCliente(rs.getLong("codi_cliente"));
          nv.setCodiVendedor(rs.getLong("codi_vendedor"));
          nv.setObservacion(rs.getString("observacion"));
          nv.setSubTotal(rs.getDouble("subtotal"));
          nv.setTotal(rs.getDouble("total"));
          nv.setTotalDcto(rs.getDouble("total_dcto"));
          nv.setEstado(rs.getString("estado"));
          nv.setFecha_registro(rs.getDate("fecha_registro"));
          nombre = rs.getString("nombre");
          apellido = rs.getString("apellido");
          nv.setNombreVendedor(nombre+" "+apellido);
          nombre = rs.getString("nombre1");
          apellido = rs.getString("apellido1");
          nv.setNombreCliente(nombre+" "+apellido);
        }

        if (rs != null)
          rs.close();
```

```
        con.disconnect();
        return nv;
    }


    public ArrayList listarNotaCredito(int criterio) throws SQLException,
Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        String queryString = "";
        DBConectionDAO con = new DBConectionDAO();

        queryString = ("select * "+
                "from ven_nota_Credito f,rh_empleado_dat b, ven_cliente c "+
                "where f.codi_vendedor = b.idempleado and f.codi_cliente =
c.codi_cliente "+
                "and date_trunc('day',f.fecha_registro) >= date_trunc('day' ,
LOCALTIMESTAMP) + interval '-6 month' "+
                "and f.estado = 'A' "+
                "and b.estado = 'A' "+
                "order by codi_nota_Credito desc");

        if (criterio == 1)
            queryString = ("select * from ven_nota_Credito a, rh_empleado_dat b,
ven_cliente c where a.codi_cliente = c.codi_cliente and a.codi_vendedor =
idempleado and a.estado = 'A' and b.estado = 'A' order by codi_nota_Credito
desc");

        PreparedStatement ps = con.prepareStatement(queryString);
        rs = ps.executeQuery();
        while (rs.next()){
            NotaCredito nv= new NotaCredito();
            nv.setCodiNotaCredito(rs.getLong("codi_nota_Credito"));
            nv.setCodiTipoCliente(rs.getLong("codi_cliente"));
            nv.setCodiVendedor(rs.getLong("codi_vendedor"));
            nv.setObservacion(rs.getString("observacion"));
            nv.setSubTotal(rs.getDouble("subtotal"));
            nv.setTotal(rs.getDouble("total"));
            nv.setTotalDcto(rs.getDouble("total_dcto"));
            nv.setEstado(rs.getString("estado"));
            nv.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
```

```
            apellido = rs.getString("apellido");
            nv.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            nv.setNombreCliente(nombre+" "+apellido);
            lista.add(nv);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarNotaCreditoFecha(String fecha1, String fecha2)
throws SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select
a.*,b.nombre,b.apellido,c.nombre1,c.apellido1 from ven_nota_credito a,
rh_empleado_dat b, ven_cliente c where a.codi_cliente = c.codi_cliente and
a.codi_vendedor = idempleado and a.estado = 'A' and b.estado = 'A' and
date_trunc('day',a.fecha_registro) between ? and ? order by
codi_nota_Credito desc");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setString(1,fecha1);
        ps.setString(2,fecha2);
        rs = ps.executeQuery();
        while (rs.next()){
            NotaCredito nv= new NotaCredito();
            nv.setCodiNotaCredito(rs.getLong("codi_nota_Credito"));
            nv.setCodiTipoCliente(rs.getLong("codi_cliente"));
            nv.setCodiVendedor(rs.getLong("codi_vendedor"));
            nv.setObservacion(rs.getString("observacion"));
            nv.setSubTotal(rs.getDouble("subtotal"));
            nv.setTotal(rs.getDouble("total"));
            nv.setEstado(rs.getString("estado"));
            nv.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            nv.setNombreVendedor(nombre+" "+apellido);
```

```
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            nv.setNombreCliente(nombre+" "+apellido);
            lista.add(nv);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarNotaCreditoMonto(double montoi, double montof)
throws SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_nota_Credito a,
rh_empleado_dat b, ven_cliente c where a.codi_cliente = c.codi_cliente and
a.codi_vendedor = idempleado and a.estado = 'A' and b.estado = 'A' and total
between ? and ? order by total;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setDouble(1,montoi);
        ps.setDouble(2,montof);
        rs = ps.executeQuery();
        while (rs.next()){
            NotaCredito f= new NotaCredito();
            f.setCodiNotaCredito(rs.getLong("codi_nota_Credito"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setTotal(rs.getDouble("total"));

            f.setEstado(rs.getString("estado"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreVendedor(nombre+" "+apellido);
```

```
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarNotaCreditoVendedor(long vendedor) throws
SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_nota_Credito a,
rh_empleado_dat b, ven_cliente c where a.codi_cliente = c.codi_cliente and
a.codi_vendedor = idempleado and a.estado = 'A' and b.estado = 'A' and
codi_vendedor = ? order by codi_nota_Credito desc;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setDouble(1,vendedor);
        rs = ps.executeQuery();
        while (rs.next()){
            NotaCredito f= new NotaCredito();
            f.setCodiNotaCredito(rs.getLong("codi_nota_Credito"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setTotal(rs.getDouble("total"));

            f.setEstado(rs.getString("estado"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
```

```
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList listarNotaCreditoCliente(String criterio) throws
SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        String nombre = null;
        String apellido = null;
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_nota_Credito a, ven_cliente b,
rh_empleado_dat c where a.codi_cliente = b.codi_cliente and c.idempleado =
a.codi_vendedor and upper(b.apellido1) like upper(?) and a.estado = 'A' and
b.estado = 'A' order by codi_nota_Credito desc;");
        PreparedStatement ps = con.prepareStatement(queryString);
        criterio+="%";
        ps.setString(1,criterio);
        rs = ps.executeQuery();
        while (rs.next()){
            NotaCredito f= new NotaCredito();
            f.setCodiNotaCredito(rs.getLong("codi_nota_Credito"));
            f.setCodiTipoCliente(rs.getLong("codi_cliente"));
            f.setCodiVendedor(rs.getLong("codi_vendedor"));
            f.setObservacion(rs.getString("observacion"));
            f.setSubTotal(rs.getDouble("subtotal"));
            f.setTotal(rs.getDouble("total"));

            f.setEstado(rs.getString("estado"));
            f.setFecha_registro(rs.getDate("fecha_registro"));
            nombre = rs.getString("nombre");
            apellido = rs.getString("apellido");
            f.setNombreVendedor(nombre+" "+apellido);
            nombre = rs.getString("nombre1");
            apellido = rs.getString("apellido1");
            f.setNombreCliente(nombre+" "+apellido);
            lista.add(f);
        }

        if (rs != null)
```

```
            rs.close();
        con.disconnect();
        return lista;
    }

    public ArrayList buscaDetalle(long id_notaCredito) throws SQLException,
Exception {
        ResultSet rs = null;
        DetalleNotaCredito dnv;
        ArrayList lista = new ArrayList();
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_detalle_nota_Credito where
codi_nota_Credito = ?;");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setLong(1,id_notaCredito);
        rs = ps.executeQuery();
        while (rs.next()){
            dnv = new DetalleNotaCredito();
            dnv.setCodiNotacredito(rs.getLong("codi_nota_Credito"));

dnv.setCodiDetallenotacredito(rs.getLong("codi_detalle_nota_Credito"));
            dnv.setCodiArti(rs.getLong("codi_arti"));
            dnv.setCantidad(rs.getLong("cantidad"));
            dnv.setPrecio(rs.getDouble("precio"));
            dnv.setPrecioParaComision(rs.getDouble("precio_para_comision"));
            dnv.setTotal(rs.getDouble("total"));
            dnv.setTotalParaComision(rs.getDouble("total_para_comision"));
            lista.add(dnv);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }
}
```

VENDEDOR DAO

```
package com.ug.cisc.erp.ventas.dao;


import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import com.ug.cisc.erp.ventas.entity.*;

public class VendedorDAO {
    String error;
    public VendedorDAO() {
        super();
        // TODO Auto-generated constructor stub
    }

    public void guardar(Vendedor vend)
    throws SQLException, Exception {
        DBConectionDAO con = new DBConectionDAO();
        try {
            con.setAutoCommit(false);
            PreparedStatement ps;

            ps = con.prepareStatement("Insert into ven_vendedor
(codi_vendedor, codi_comision, codi_vendedor_tipo, estado, fecha_creacion)
values (?,?,?,'A',date_trunc('second' , LOCALTIMESTAMP))");

            ps.setLong(1, vend.getCodiVendedor());
            ps.setLong(2, vend.getCodiComision());
            ps.setLong(3, vend.getCodiVendedorTipo());

            ps.execute();


            con.commit();

        } catch (SQLException sqle) {
            con.rollback();
            error = sqle.toString();//"SQLException: update failed, possible
duplicate entry";
            throw new SQLException(error);
```

```
        } finally {
            con.disconnect();
        }
    }

    public ArrayList muestratipos() throws SQLException, Exception {
        ResultSet rs = null;
        ArrayList lista = new ArrayList();
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from ven_vendedor_tipo where estado =
'A'");
        PreparedStatement ps = con.prepareStatement(queryString);
        rs = ps.executeQuery();
        while (rs.next()){
            VendedorTipo c= new VendedorTipo();
            c.setVendedorTipo(rs.getString("vendedor_tipo"));
            c.setDescripcion(rs.getString("descripcion"));
            c.setCodiVendedorTipo(rs.getLong("codi_vendedor_tipo"));
            lista.add(c);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

    public Vendedor buscaempleado(long vend) throws SQLException,
Exception {
        ResultSet rs = null;
        Vendedor v= new Vendedor();
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * from rh_empleado_dat a where estado='A'
and idempleado = ? and idempleado NOT in (select codi_vendedor from
ven_vendedor);");
        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setLong(1,vend);
        rs = ps.executeQuery();

        if (rs.next()){

            v.setNombre(rs.getString("nombre"));
            v.setApellido(rs.getString("apellido"));
            //v.setCodiVendedorTipo(rs.getLong("codi_vendedor_tipo"));
```

```
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return v;
    }

    public Vendedor buscaVendedor(long codiVend) throws SQLException,
Exception {
        ResultSet rs = null;
        Vendedor v= new Vendedor();
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * "+
                "from rh_empleado_dat a , ven_vendedor v, ven_comision vc "+
                "where a.idempleado = v.codi_vendedor "+
                "and v.codi_comision = vc.codi_comision "+
                "and v.codi_vendedor = ? "+
                "and a.estado='A' ");

        PreparedStatement ps = con.prepareStatement(queryString);
        ps.setLong(1,codiVend);
        rs = ps.executeQuery();

        if (rs.next()){
            v.setCodiVendedor(rs.getLong("codi_vendedor"));
            v.setCodiComision(rs.getLong("codi_comision"));
            v.setFechaCreacion(rs.getDate("fecha_creacion"));
            v.setCodiVendedorTipo(rs.getLong("codi_vendedor_tipo"));
            v.setNombre(rs.getString("nombre"));
            v.setApellido(rs.getString("apellido"));
            v.setTipoComision(rs.getString("tipo_comision"));
            v.setDescripcion(rs.getString("descripcion"));

        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return v;
    }
```

```
    public ArrayList buscaVendedorNombre(String criterio) throws
SQLException, Exception {
        ResultSet rs = null;
        Vendedor v= null;
        ArrayList lista = new ArrayList();
        if (criterio == null)
            criterio = "";
        DBConectionDAO con = new DBConectionDAO();
        String queryString = ("select * "+
                "from rh_empleado_dat a , ven_vendedor v, ven_comision vc "+
                "where a.idempleado = v.codi_vendedor "+
                "and v.codi_comision = vc.codi_comision "+
                "and upper(apellido) like upper(?) "+
                "and a.estado='A' ");
        PreparedStatement ps = con.prepareStatement(queryString);
        criterio+="%";
        ps.setString(1, criterio);
        rs = ps.executeQuery();

        while (rs.next()){
            v = new Vendedor();
            v.setCodiVendedor(rs.getLong("codi_vendedor"));
            v.setCodiComision(rs.getLong("codi_comision"));
            v.setFechaCreacion(rs.getDate("fecha_creacion"));
            v.setCodiVendedorTipo(rs.getLong("codi_vendedor_tipo"));
            v.setNombre(rs.getString("nombre"));
            v.setApellido(rs.getString("apellido"));
            v.setTipoComision(rs.getString("tipo_comision"));
            v.setDescripcion(rs.getString("descripcion"));
            lista.add(v);
        }

        if (rs != null)
            rs.close();
        con.disconnect();
        return lista;
    }

}
```

COMISION DAO

```
package com.ug.cisc.erp.ventas.dao;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.*;
import java.text.SimpleDateFormat;

import com.ug.cisc.erp.ventas.entity.*;

public class ComisionDAO {
  String error;

  public ComisionDAO() {
    super();
    // TODO Auto-generated constructor stub
  }


  public ArrayList muestratipos() throws SQLException, Exception {
    ResultSet rs = null;
    ArrayList lista = new ArrayList();
    String queryString = ("select * from ven_comision where estado = 'A'");

    DBConectionDAO con = new DBConectionDAO();

    PreparedStatement ps = con.prepareStatement(queryString);
    rs = ps.executeQuery();
    while (rs.next()){
      Comision c= new Comision();
      c.setTipoComision(rs.getString("tipo_comision"));
      c.setDescripcion(rs.getString("descripcion"));
      c.setCodiComision(rs.getLong("codi_comision"));
      lista.add(c);
    }
    con.disconnect();

    return lista;
  }

  public double verificaRango(double totalVenta) throws SQLException,
Exception {
```

```
    ResultSet rs = null;
    PreparedStatement ps = null;
    double porcValorComision = 0;

    DBConectionDAO con = new DBConectionDAO();
    ps = con.prepareStatement("select esnull(v.porc_valor_comision,0) as
porc_valor_comision "+
      "from ven_comision_parametro_rango v "+
      "where ? between v.rango_inicio and v.rango_fin "+
      "and v.estado = 'A' ");
    ps.setDouble(1,totalVenta);
    rs = ps.executeQuery();

    if (rs.next()){
      porcValorComision = rs.getDouble("porc_valor_comision");
    }
    con.disconnect();

    return (porcValorComision);
  }

  public double verificaRangoDev(double porcDev) throws SQLException,
Exception {
    ResultSet rs = null;
    PreparedStatement ps = null;
    double porcDevo = 0;

    DBConectionDAO con = new DBConectionDAO();
    ps = con.prepareStatement("select esnull(v.porc_dev,0) as porc_dev "+
      "from ven_comision_parametro_dev v "+
      "where ? between v.porc_rango_inicio and v.porc_rango_fin "+
      "and v.estado = 'A'");
    ps.setDouble(1,porcDev);
    rs = ps.executeQuery();

    if (rs.next()){
      porcDevo = rs.getDouble("porc_dev");
    }

    con.disconnect();
    return (porcDevo);
  }
```

```
  public double verificaRangoNVenta(double contadorTotalVentas) throws
SQLException, Exception {
    ResultSet rs = null;
    PreparedStatement ps = null;
    double porcCastigo = 0;


    DBConectionDAO con = new DBConectionDAO();
    ps = con.prepareStatement("select esnull(v.porc_castigo,0) as
porc_castigo  "+
      "from ven_comision_parametro_venta v "+
      "where ? between v.venta_rango_inicio and v.venta_rango_fin "+
      "and v.estado = 'A'");
    ps.setDouble(1,contadorTotalVentas);
    rs = ps.executeQuery();

    if (rs.next()){
      porcCastigo = rs.getDouble("porc_castigo");
    }

    con.disconnect();
    return (porcCastigo);
  }

  public double verificaRangoMeses(long codigo, String opcion) throws
SQLException, Exception {
    ResultSet rs = null;
    PreparedStatement ps = null;
    int meseMora = 0;
    double porcCastigo = 100;
    String queryString = "";

    DBConectionDAO con = new DBConectionDAO();

    if (opcion.equals("FAC"))
      queryString = "SELECT (EXTRACT(MONTH FROM
date_trunc('day',f.fecha_pago)) - EXTRACT(MONTH FROM
date_trunc('day',f.fecha_registro))) as meses_mora "+
        "FROM ven_factura f "+
        "where f.estado_control = 'P' "+
        "and f.estado = 'A' "+
        "and f.codi_factura = ? ";
    else
```

```
      queryString = "SELECT (EXTRACT(MONTH FROM
date_trunc('day',f.fecha_pago)) - EXTRACT(MONTH FROM
date_trunc('day',f.fecha_registro))) as meses_mora "+
        "FROM ven_nota_venta f "+
        "where f.estado_control = 'P' "+
        "and f.estado = 'A' "+
        "and f.codi_nota_venta = ? ";

    ps = con.prepareStatement(queryString);

    ps.setLong(1,codigo);
    rs = ps.executeQuery();

    if (rs.next()){
      meseMora = rs.getInt("meses_mora");
    }

    ps = con.prepareStatement("select v.porc_castigo "+
      "from ven_comision_parametro_fecha v "+
      "where v.numero_meses = ? "+
      "and v.estado = 'A'");

    ps.setLong(1,meseMora);
    rs = ps.executeQuery();

    if (rs.next()){
      porcCastigo = rs.getDouble("porc_castigo");
    }

    con.disconnect();

    return (porcCastigo);
  }

  public void calculaPorFechaCobroFactura(ArrayList fac) throws
SQLException, Exception {
    Factura factura = null;
    Iterator ifac = fac.iterator();
    FacturaDAO facturaDAO = new FacturaDAO();
    double total = 0;
    long codiFactura = 0;
    double porcCastigo = 0;
    double totalPenalizado = 0;
    double valorPenalizacion = 0;
```

```
      while (ifac.hasNext()){
        Factura f = (Factura)ifac.next();
        total = f.getSubTotalPenalizadoParaComision();
        codiFactura = f.getCodiFactura();
        porcCastigo = verificaRangoMeses(codiFactura, "FAC");
        valorPenalizacion  = (total * porcCastigo)/100;
        totalPenalizado = total - valorPenalizacion ;

        factura = new Factura();
        factura.setCodiFactura(codiFactura);
        factura.setSubTotalPenalizadoParaComision(totalPenalizado);
        facturaDAO.actualizaCabecera(factura);
      }

  }

  public void calculaPorFechaCobroNVenta(ArrayList nve) throws
SQLException, Exception {
    NotaVenta notaVenta = null;
    Iterator inve = nve.iterator();
    NotaVentaDAO notaVentaDAO = new NotaVentaDAO();
    double total = 0;
    long codiNVenta = 0;
    double porcCastigo = 0;
    double totalPenalizado = 0;
    double valorPenalizacion = 0;

      while (inve.hasNext()){
        NotaVenta n = (NotaVenta)inve.next();
        total = n.getSubTotalPenalizadoParaComision();
        codiNVenta = n.getCodiNotaVenta();

        porcCastigo = verificaRangoMeses(codiNVenta,"NVEN");
        valorPenalizacion  = (total * porcCastigo)/100;
        totalPenalizado = total - valorPenalizacion ;

        notaVenta = new NotaVenta();
        notaVenta.setCodiNotaVenta(codiNVenta);
        notaVenta.setSubTotalPenalizadoParaComision(totalPenalizado);
        notaVentaDAO.actualizaCabecera(notaVenta);
      }

  }
```

```
  public VendedorBono calculaPorDiferencia(long codi_vendedor)throws
SQLException, Exception {
    ResultSet rs = null;
    String queryString = "";
    PreparedStatement ps = null;
    java.util.Date fechaActual = new java.util.Date();
    SimpleDateFormat formato = new SimpleDateFormat("MM");
    String mesComision = formato.format(fechaActual);
    VendedorBono vendedorBono = new VendedorBono();
    long contadorFac = 0;
    long contadorNotaVenta = 0;
    long contadorTotalVentas = 0;
    double totalFac = 0;
    double totalFacParaComision = 0;
    double totalNov = 0;
    double totalNovParaComision = 0;
    double totalNoc = 0;
    double totalNocParaComision = 0;
    double total_venta = 0;
    double total_venta_neta = 0;
    double total_devolucion = 0;
    double total_venta_com = 0;
    double total_venta_neta_com = 0;
    double total_devolucion_com = 0;
    //double total_comision = 0;
    double mitadComision = 0;
    double comision = 0;
    //double porcRango = 0;
    double porcDev = 0;
    double porcPenalizaPorDev = 0;
    double porcPenalizaPorNVentas = 0;
    double porcPenaliza = 0;
    double valorPenaliza = 0;
    double totalComisionPagar = 0;
    String observacion = "";
    Factura fac = null;
    NotaVenta nve = null;
    ArrayList listaFac = new ArrayList();
    ArrayList listaNve = new ArrayList();

    DBConectionDAO con = new DBConectionDAO();
```

```
    queryString = ("select
fac.codi_factura,(esnull(fac.subtotal_para_comision,0) - esnull(fac.subtotal,0))
as total_para_comision "+
      "from ven_factura fac "+
      "where fac.estado_control = 'P' "+
      "and fac.estado = 'A' "+
      "and date_trunc('day',fac.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and fac.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    while (rs.next()){
      //totalFac = rs.getDouble("total");
      fac = new Factura();
      fac.setCodiFactura(rs.getLong("codi_factura"));

fac.setSubTotalPenalizadoParaComision(rs.getDouble("total_para_comision"
));
      listaFac.add(fac);
      //contadorFac = rs.getLong("n_registros");
    }

    calculaPorFechaCobroFactura(listaFac);

    queryString = ("select
notv.codi_nota_venta,(esnull(notv.subtotal_para_comision,0) -
esnull(notv.subtotal,0)) as total_para_comision "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A'  "+
      "and notv.estado_control = 'P' "+
      "and date_trunc('day',notv.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    while (rs.next()){
      //totalNov = rs.getDouble("total");
      nve = new NotaVenta();
```

```
      nve.setCodiNotaVenta(rs.getLong("codi_nota_venta"));

nve.setSubTotalPenalizadoParaComision(rs.getDouble("total_para_comision"
));
      listaNve.add(nve);
      //contadorNotaVenta = rs.getLong("n_registros");
    }

    calculaPorFechaCobroNVenta(listaNve);

    // Sumatoria factura
    queryString = ("select sum(esnull(fac.subtotal,0)) as
total,sum(esnull(fac.subtotal_penalizado_para_comision,0)) as
total_para_comision "+
      "from ven_factura fac "+
      "where fac.estado = 'A' "+
      "and date_trunc('day',fac.fecha_registro) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and fac.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next()){
      totalFac = rs.getDouble("total");
      totalFacParaComision = rs.getDouble("total_para_comision");
    }

    // Sumatoria Nota Venta
    queryString = ("select sum(esnull(notv.subtotal,0)) as
total,sum(esnull(notv.subtotal_penalizado_para_comision,0)) as
total_para_comision "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A' "+
      "and date_trunc('day',notv.fecha_registro) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next()){
```

```
      totalNov = rs.getDouble("total");
      totalNovParaComision = rs.getDouble("total_para_comision");
    }

    // Sumatoria Nota Credito
    queryString = ("select sum(esnull(notc.subtotal,0)) as
total,sum(esnull(notc.subtotal_para_comision,0) - esnull(notc.subtotal,0)) as
total_para_comision "+
      "from ven_nota_credito notc "+
      "where notc.estado = 'A' "+
      "and date_trunc('day',notc.fecha_registro) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notc.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next()){
      totalNoc = rs.getDouble("total");
      totalNocParaComision = rs.getDouble("total_para_comision");
    }



    queryString = ("select count (*) as n_registros "+
      "from ven_factura fac "+
      "where fac.estado_control = 'P' "+
      "and fac.estado = 'A' "+
      "and date_trunc('day',fac.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and fac.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next())
      contadorFac = rs.getLong("n_registros");

    queryString = ("select count (*) as n_registros "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A'  "+
      "and notv.estado_control = 'P' "+
```

```
      "and date_trunc('day',notv.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next())
      contadorNotaVenta = rs.getLong("n_registros");

    contadorTotalVentas = contadorFac + contadorNotaVenta;

                            /*total_venta = totalFac + totalNov;
                            total_devolucion = totalNoc;
                            total_venta_neta = total_venta - total_devolucion;*/

    total_venta = totalFac + totalNov;
    total_devolucion = totalNoc;
    total_venta_neta = total_venta - total_devolucion;

    total_venta_com = totalFacParaComision + totalNovParaComision;
    total_devolucion_com = totalNocParaComision;
    total_venta_neta_com = total_venta_com - total_devolucion_com;

                            /*
                             * Se verifica Castigo por devoluciones y/o por numero de
ventas
                             * */

    //porcRango = verificaRango(total_venta_neta_com);
    //comision = (total_venta_neta_com * porcRango) / 100;
    comision = total_venta_neta_com;

    mitadComision = comision / 2; // El 50% entra a verificacion de castigo
    if (total_venta_com != 0)
      porcDev = (total_devolucion_com * 100) / total_venta_com;
    else
      porcDev = 0;

    porcPenalizaPorDev = verificaRangoDev(porcDev);
    porcPenalizaPorNVentas = verificaRangoNVenta(contadorTotalVentas);

    if (porcPenalizaPorDev > 0)
```

```
    observacion = "Monto de Devolución \n";

  if (porcPenalizaPorNVentas > 0)
    observacion += "Número de Ventas";

  porcPenaliza = (porcPenalizaPorDev + porcPenalizaPorNVentas)/ 2;
  valorPenaliza = (mitadComision * porcPenaliza) / 100;

  totalComisionPagar = mitadComision - valorPenaliza;
  totalComisionPagar = totalComisionPagar + mitadComision;

  //total_comision = total_venta_neta_com - total_venta_neta;

  //VendedorBono vendedorBono = new VendedorBono();
  vendedorBono.setCodiVendedor(codi_vendedor);
  vendedorBono.setValorVenta(total_venta);
  vendedorBono.setValorComision(comision);
  vendedorBono.setValorComisionPagar(totalComisionPagar);
  vendedorBono.setMesComision(mesComision);
  vendedorBono.setValorDevolucion(total_devolucion);
  vendedorBono.setNumeroVenta(contadorTotalVentas);
  vendedorBono.setObservacion(observacion);
  vendedorBono.setEstado("A");

  if (rs!= null)
     rs.close();
  con.disconnect();

  return (vendedorBono);
 }

 public VendedorBono calculaPorPorcentaje(long codi_vendedor)throws
SQLException, Exception {
  ResultSet rs = null;
  String queryString = "";
  PreparedStatement ps = null;
  java.util.Date fechaActual = new java.util.Date();
  SimpleDateFormat formato = new SimpleDateFormat("MM");
  String mesComision = formato.format(fechaActual);
  VendedorBono vendedorBono = new VendedorBono();
  long contadorFac = 0;
  long contadorNotaVenta = 0;
  long contadorTotalVentas = 0;
  double totalFac = 0;
```

```
double totalFacParaComision = 0;
double totalPenFacParaComision = 0;
double totalNov = 0;
double totalNovParaComision = 0;
double totalPenNovParaComision = 0;
//double totalNoc = 0;
double totalNocParaComision = 0;
//double total_venta = 0;
//double total_devolucion = 0;
double total_venta_com = 0;
double total_venta_neta_com = 0;
double total_devolucion_com = 0;
double mitadComision = 0;
double comision = 0;
double porcRango = 0;
double porcDev = 0;
double porcPenalizaPorDev = 0;
double porcPenalizaPorNVentas = 0;
//double porcPenalizaPorMesesMora = 0;
double porcPenaliza = 0;
double valorPenaliza = 0;
double totalComisionPagar = 0;
String observacion = "";
Factura fac = null;
NotaVenta nve = null;
ArrayList listaFac = new ArrayList();
ArrayList listaNve = new ArrayList();


DBConectionDAO con = new DBConectionDAO();
queryString = ("select sum(esnull(fac.subtotal_para_comision,0)) as
total_para_comision "+
  "from ven_factura fac "+
  "where fac.estado_control = 'P' "+
  "and fac.estado = 'A' "+
  "and date_trunc('day',fac.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
  "and fac.codi_vendedor = ? ");

ps = con.prepareStatement(queryString);
ps.setLong(1,codi_vendedor);
rs = ps.executeQuery();

if (rs.next()){
```

```
      //totalFac = rs.getDouble("total");
      totalFacParaComision = rs.getDouble("total_para_comision");
      //contadorFac = rs.getLong("n_registros");
    }

    queryString = ("select sum(esnull(notv.subtotal_para_comision,0)) as
total_para_comision "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A'  "+
      "and notv.estado_control = 'P' "+
      "and date_trunc('day',notv.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next()){
      //totalNov = rs.getDouble("total");
      totalNovParaComision = rs.getDouble("total_para_comision");
      //contadorNotaVenta = rs.getLong("n_registros");
    }

    queryString = ("select sum(esnull(notc.subtotal_para_comision,0)) as
total_para_comision "+
      "from ven_nota_credito notc "+
      "where notc.estado = 'A' "+
      "and date_trunc('day',notc.fecha_registro) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notc.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next()){
      //totalNoc = rs.getDouble("total");
      totalNocParaComision = rs.getDouble("total_para_comision");
    }

    queryString = ("select count (*) as n_registros "+
      "from ven_factura fac "+
      "where fac.estado_control = 'P' "+
```

```
      "and fac.estado = 'A' "+
      "and date_trunc('day',fac.fecha_pago) > 
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and fac.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next())
      contadorFac = rs.getLong("n_registros");

    queryString = ("select count (*) as n_registros "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A'  "+
      "and notv.estado_control = 'P' "+
      "and date_trunc('day',notv.fecha_pago) > 
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next())
      contadorNotaVenta = rs.getLong("n_registros");

    contadorTotalVentas = contadorFac + contadorNotaVenta;

    //total_venta = totalFac + totalNov;
    //total_devolucion = totalNoc;
    //total_venta_neta = total_venta - total_devolucion;


    total_venta_com = totalFacParaComision + totalNovParaComision;
    total_devolucion_com = totalNocParaComision;
    total_venta_neta_com = total_venta_com - total_devolucion_com;

                          /*
                           * Se verifica Castigo por devoluciones y/o por numero de 
ventas
                           * */

    porcRango = verificaRango(total_venta_neta_com);
```

```
    queryString = ("select
fac.codi_factura,esnull(fac.subtotal_para_comision,0) as total_para_comision
"+
      "from ven_factura fac "+
      "where fac.estado_control = 'P' "+
      "and fac.estado = 'A' "+
      "and date_trunc('day',fac.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and fac.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    while (rs.next()){
      totalFac = rs.getDouble("total_para_comision");
      totalFac = (totalFac * porcRango)/100;
      fac = new Factura();
      fac.setCodiFactura(rs.getLong("codi_factura"));
      fac.setSubTotalPenalizadoParaComision(totalFac);
      listaFac.add(fac);
      //contadorFac = rs.getLong("n_registros");
    }

    calculaPorFechaCobroFactura(listaFac);

    queryString = ("select
notv.codi_nota_venta,esnull(notv.subtotal_para_comision,0) as
total_para_comision "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A'  "+
      "and notv.estado_control = 'P' "+
      "and date_trunc('day',notv.fecha_pago) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();



    while (rs.next()){
```

```
      totalNov = rs.getDouble("total_para_comision");
      totalNov = (totalNov * porcRango)/100;
      nve = new NotaVenta();
      nve.setCodiNotaVenta(rs.getLong("codi_nota_venta"));
      nve.setSubTotalPenalizadoParaComision(totalNov);
      listaNve.add(nve);
      //contadorNotaVenta = rs.getLong("n_registros");
    }

    calculaPorFechaCobroNVenta(listaNve);

    // Sumatoria factura
    queryString = ("select
sum(esnull(fac.subtotal_penalizado_para_comision,0)) as
total_penalizado_para_comision "+
      "from ven_factura fac "+
      "where fac.estado = 'A' "+
      "and date_trunc('day',fac.fecha_registro) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and fac.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();

    if (rs.next()){
      //totalFac = rs.getDouble("total");
      totalPenFacParaComision =
rs.getDouble("total_penalizado_para_comision");
    }

    // Sumatoria Nota Venta
    queryString = ("select
sum(esnull(notv.subtotal_penalizado_para_comision,0)) as
total_penalizado_para_comision "+
      "from ven_nota_venta notv "+
      "where notv.estado = 'A' "+
      "and date_trunc('day',notv.fecha_registro) >
date_trunc('day',LOCALTIMESTAMP) + interval '-1 month' "+
      "and notv.codi_vendedor = ? ");

    ps = con.prepareStatement(queryString);
    ps.setLong(1,codi_vendedor);
    rs = ps.executeQuery();
```

```
    if (rs.next()){
      //totalNov = rs.getDouble("total");
      totalPenNovParaComision =
rs.getDouble("total_penalizado_para_comision");
    }

    //comision = (total_venta_neta_com * porcRango) / 100;
    comision = totalPenFacParaComision + totalPenNovParaComision;
    mitadComision = comision / 2; // El 50% entra a verificacion de castigo


    if (total_venta_com != 0)
      porcDev = (total_devolucion_com * 100) / total_venta_com;
    else
      porcDev = 0;

    porcPenalizaPorDev = verificaRangoDev(porcDev);
    porcPenalizaPorNVentas = verificaRangoNVenta(contadorTotalVentas);
    //porcPenalizaPorMesesMora = verificaRangoMeses(codi_vendedor);

    if (porcPenalizaPorDev > 0)
      observacion = "Monto de Devolución \n";

    if (porcPenalizaPorNVentas > 0)
      observacion += "Número de Ventas";

    porcPenaliza = porcPenalizaPorDev + porcPenalizaPorNVentas;
    valorPenaliza = (mitadComision * porcPenaliza) / 100;

    totalComisionPagar = mitadComision - valorPenaliza;
    totalComisionPagar = totalComisionPagar + mitadComision;

    vendedorBono.setCodiVendedor(codi_vendedor);
    vendedorBono.setValorVenta(total_venta_neta_com);
    vendedorBono.setValorComision(comision);
    vendedorBono.setValorComisionPagar(totalComisionPagar);
    vendedorBono.setMesComision(mesComision);
    vendedorBono.setValorDevolucion(total_devolucion_com);
    vendedorBono.setNumeroVenta(contadorTotalVentas);
    vendedorBono.setObservacion(observacion);
    vendedorBono.setEstado("A");

    con.disconnect();
```

```
    return (vendedorBono);
  }


  public void calculaComisionPorDiferencia() throws SQLException, Exception
{

    VendedorBonoDAO vendedorBonoDAO = new VendedorBonoDAO();
    ArrayList listaVendedor = null;
    Iterator contents = null;
    //VendedorBono vendedorBono = new VendedorBono();

    try  {

      listaVendedor = vendedorBonoDAO.listarVendedor("DIF"); // lista de
ven_vendedor
      contents = listaVendedor.iterator();

      while (contents.hasNext()){
        Vendedor vendedor = (Vendedor)contents.next();
        VendedorBono vendedorBono =
calculaPorDiferencia(vendedor.getCodiVendedor());
        if
(vendedorBonoDAO.buscaVendedorBono(vendedor.getCodiVendedor()) ==
null){
          vendedorBonoDAO.inserta(vendedorBono);
        }else{
          vendedorBonoDAO.actualiza(vendedorBono);
        }
        vendedorBonoDAO.insertaBitacora(vendedorBono);
      }

    } catch (Exception e) {
      error = "Ha ocurrido una excepcion consultando la base de datos.
Contactese con el Webmaster";
      throw new Exception(error);
    }finally{
      //disconnect();

    }

  }
```

```
  public void calculaComisionPorFechaCobro() throws SQLException,
Exception {

    VendedorBonoDAO vendedorBonoDAO = new VendedorBonoDAO();
    FacturaDAO facturaDAO = new FacturaDAO();
    NotaVentaDAO notaVentaDAO = new NotaVentaDAO();
    ArrayList listaVendedor = null;
    ArrayList listaFactura = null;
    ArrayList listaNVenta = null;
    Iterator contents = null;
    Iterator ilistaFactura = null;
    Iterator ilistaNVenta = null;

    try  {

      listaVendedor = vendedorBonoDAO.listarVendedor(""); // lista de
ven_vendedor
      contents = listaVendedor.iterator();
      while (contents.hasNext()){
        Vendedor vendedor = (Vendedor)contents.next();
        //listaFactura =
calculaPorFechaCobroFactura(vendedor.getCodiVendedor());
        ilistaFactura = listaFactura.iterator();

        while (ilistaFactura.hasNext()){
          Factura factura = (Factura)ilistaFactura.next();
          facturaDAO.actualizaCabecera(factura);
        }

      }


      listaVendedor = vendedorBonoDAO.listarVendedor(""); // lista de
ven_vendedor
      contents = listaVendedor.iterator();
      while (contents.hasNext()){
        Vendedor vendedor = (Vendedor)contents.next();
        //listaNVenta =
calculaPorFechaCobroNVenta(vendedor.getCodiVendedor());
        ilistaNVenta = listaNVenta.iterator();

        while (ilistaNVenta.hasNext()){
          NotaVenta notaVenta = (NotaVenta)ilistaNVenta.next();
          notaVentaDAO.actualizaCabecera(notaVenta);
```

```
        }

      }

    } catch (Exception e) {
      error = "Ha ocurrido una excepcion consultando la base de datos.
Contactese con el Webmaster";
      throw new Exception(error);
    }

  }

  public void calculaComisionPorPorcentaje() throws SQLException,
Exception {

    VendedorBonoDAO vendedorBonoDAO = new VendedorBonoDAO();
    ArrayList listaVendedor = null;
    Iterator contents = null;

    try  {

      listaVendedor = vendedorBonoDAO.listarVendedor("POR"); // lista de
ven_vendedor
      contents = listaVendedor.iterator();
      while (contents.hasNext()){
        Vendedor vendedor = (Vendedor)contents.next();
        VendedorBono vendedorBono =
calculaPorPorcentaje(vendedor.getCodiVendedor());
        if
(vendedorBonoDAO.buscaVendedorBono(vendedor.getCodiVendedor()) ==
null){
          vendedorBonoDAO.inserta(vendedorBono);
        }else{
          vendedorBonoDAO.actualiza(vendedorBono);
        }
        vendedorBonoDAO.insertaBitacora(vendedorBono);
      }

    } catch (Exception e) {
      error = "Ha ocurrido una excepcion consultando la base de datos.
Contactese con el Webmaster";
      throw new Exception(error);
    }
```

```
  }

  public void guardar(Comision comi)
  throws SQLException, Exception {
    DBConectionDAO con = new DBConectionDAO();
    try {
      con.setAutoCommit(false);
      PreparedStatement ps;

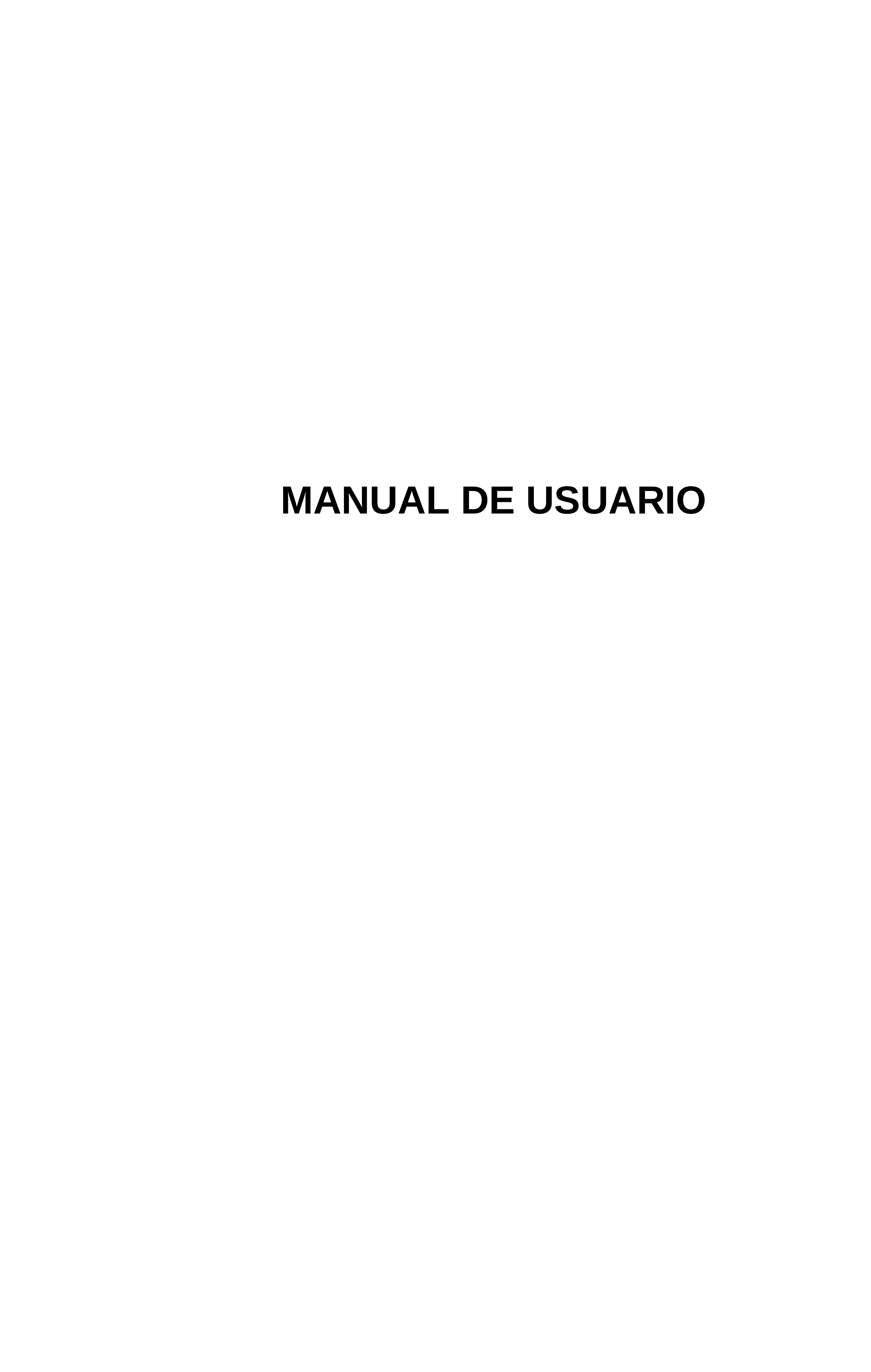
      ps = con.prepareStatement("Insert into ven_comision (tipo_comision,
descripcion, estado) values (?,?,'A')");

      ps.setString(1, comi.getTipoComision());
      ps.setString(2, comi.getDescripcion());
      ps.execute();
      con.commit();

    } catch (SQLException sqle) {
      con.rollback();
      error = sqle.toString();//"SQLException: update failed, possible duplicate
entry";
      throw new SQLException(error);
    }
  }

  public void calculaComision() throws SQLException, Exception {
    try  {
      //calculaComisionPorFechaCobro();
      calculaComisionPorDiferencia();
      calculaComisionPorPorcentaje();
    } catch (Exception e) {
      error = "Ha ocurrido una excepcion consultando la base de datos.
Contactese con el Webmaster";
      throw new Exception(error);
    }
  }

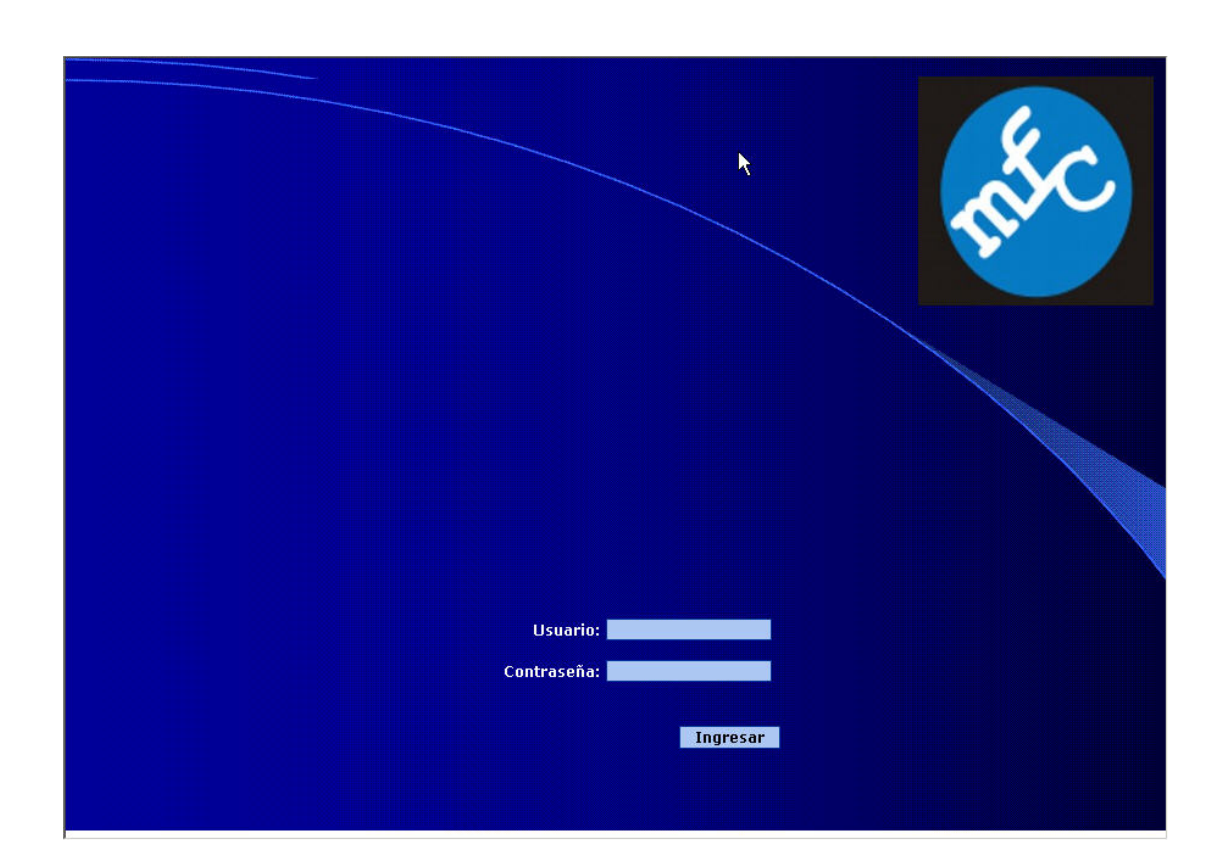
}
```



MANUAL DE USUARIO

MÓDULO DE FACTURACIÓN Y COMISIONES

Para ingresar al módulo de ventas y comisiones es necesario que sea un usuario registrado ya sea como administrador, vendedor senior, vendedor junior o invitado.



En la pantalla principal del módulo de ventas podrá encontrar en la sección izquierda los menús para realizar la facturación, las notas de venta, las proformas, las notas de crédito, así como también las opciones disponibles para crear nuevos vendedores calcular comisiones y elaborar reportes.



FACTURACIÓN



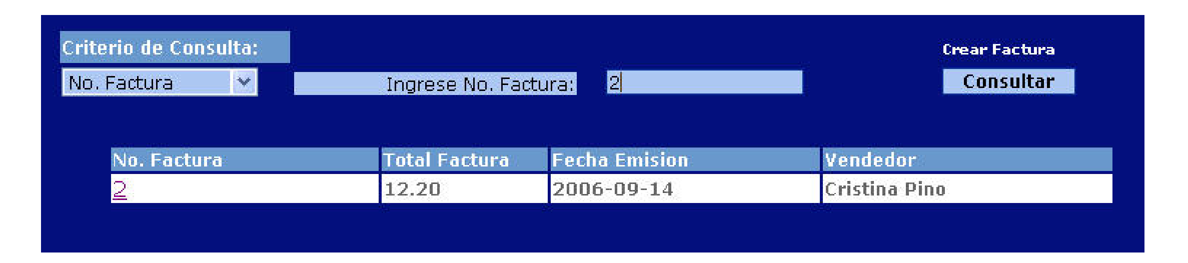
No. Factura	Total Factura	Fecha Emision	Vendedor	Cliente
14	16.27	2006-10-08	Gabriel Gallo	Vladimir Palacios
13	87.56	2006-10-07	Cristina Pino	Vladimir Palacios
12	113.01	2006-10-07	Gabriel Gallo	Vladimir Palacios
11	285.44	2006-10-07	Cristina Pino	Geovanny Reyes
10	51.37	2006-10-07	Gabriel Gallo	Geovanny Reyes
9	13.40	2006-10-07	Cristina Pino	Francisco Triviño
8	13.05	2006-10-07	Gabriel Gallo	Francisco Triviño
7	13.40	2006-10-07	Gabriel Gallo	Francisco Triviño
6	13.05	2006-10-07	Cristina Pino	Vladimir Palacios
5	120.50	2006-10-07	Gabriel Gallo	Geovanny Reyes
4	1630.77	2006-10-07	Cristina Pino	Vladimir Palacios
3	981.44	2006-10-07	Cristina Pino	Francisco Triviño
2	1356.90	2006-10-07	Cristina Pino	Francisco Triviño
1	204.28	2006-10-07	Cristina Pino	Vladimir Palacios

Consulta de Facturas

Accediendo a esta sección del programa puede visualizar de forma directa las facturas que han sido elaboradas durante el mes corriente.

En la parte superior puede seleccionar el criterio de consulta más conveniente de acuerdo a sus necesidades:

Por número de factura. Cuando selecciona esta opción se habilita un campo donde puede ingresar el número correspondiente a la factura que desea consultar.

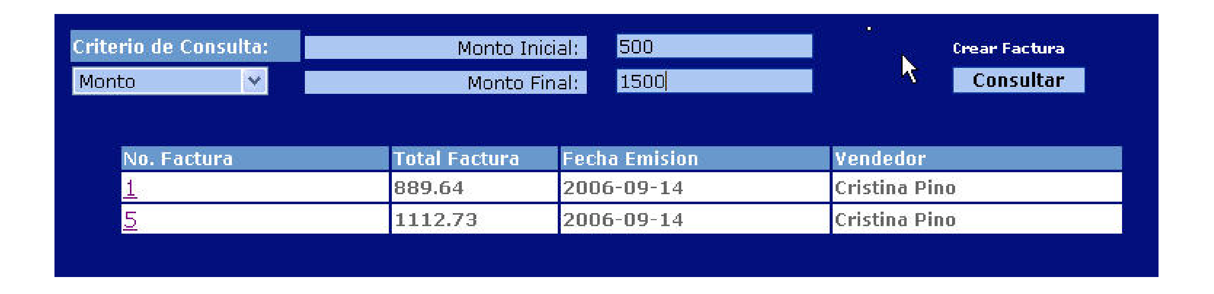


Por cliente. Al seleccionar esta opción, puede consultar de acuerdo al nombre del cliente al que se le hizo una venta.

Criterio de Consulta:
Cliente
Ingrese Nombre Cliente: Geovanny Reyes
Crear Factura
Consultar

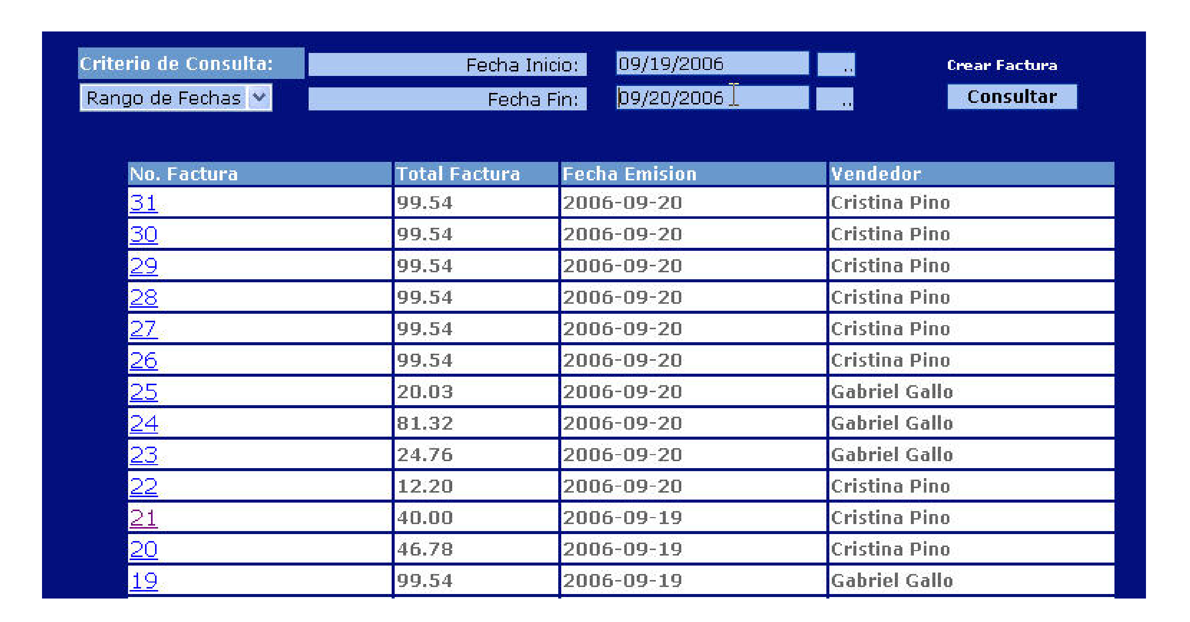
No. Factura	Total Factura	Fecha Emision	Vendedor	Cliente
31	99.54	2006-09-20	Cristina Pino	Geovanny Reyes
30	99.54	2006-09-20	Cristina Pino	Geovanny Reyes
29	99.54	2006-09-20	Cristina Pino	Geovanny Reyes
28	99.54	2006-09-20	Cristina Pino	Geovanny Reyes
27	99.54	2006-09-20	Cristina Pino	Geovanny Reyes
26	99.54	2006-09-20	Cristina Pino	Geovanny Reyes
20	46.78	2006-09-19	Cristina Pino	Geovanny Reyes

Por monto: Aquí puede el usuario ingresar un monto inicial y un monto final para realizar la consulta de todas las facturas cuyos totales estén comprendidos en este rango.



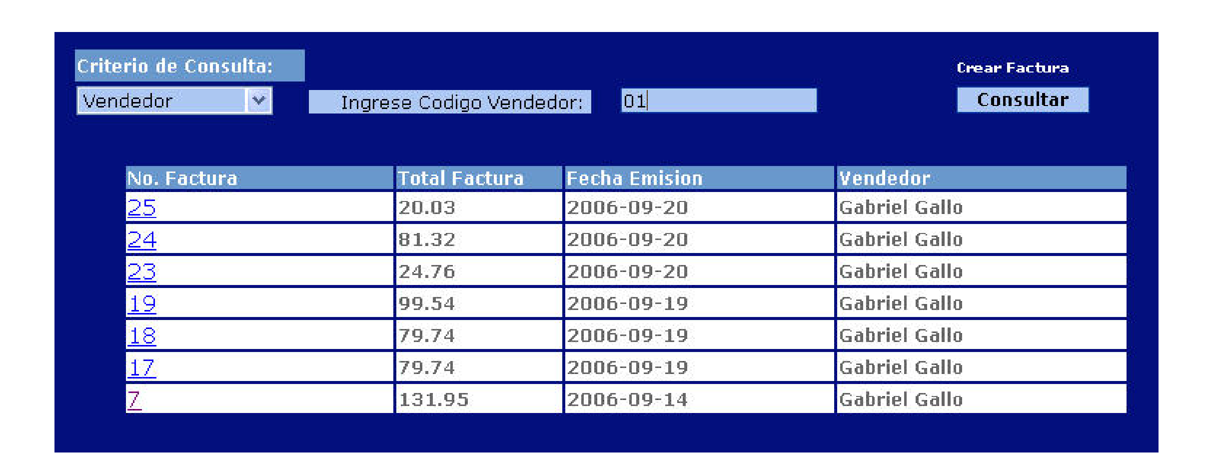
No. Factura	Total Factura	Fecha Emision	Vendedor
1	889.64	2006-09-14	Cristina Pino
5	1112.73	2006-09-14	Cristina Pino

Por fechas: El usuario podrá ingresar una fecha inicial y una fecha final para consultar todas las facturas que se hayan realizado en un período determinado. Puedo optar por seleccionar la fecha deseada directamente del calendario presionando en los botones que se encuentran a lado de la caja de texto.



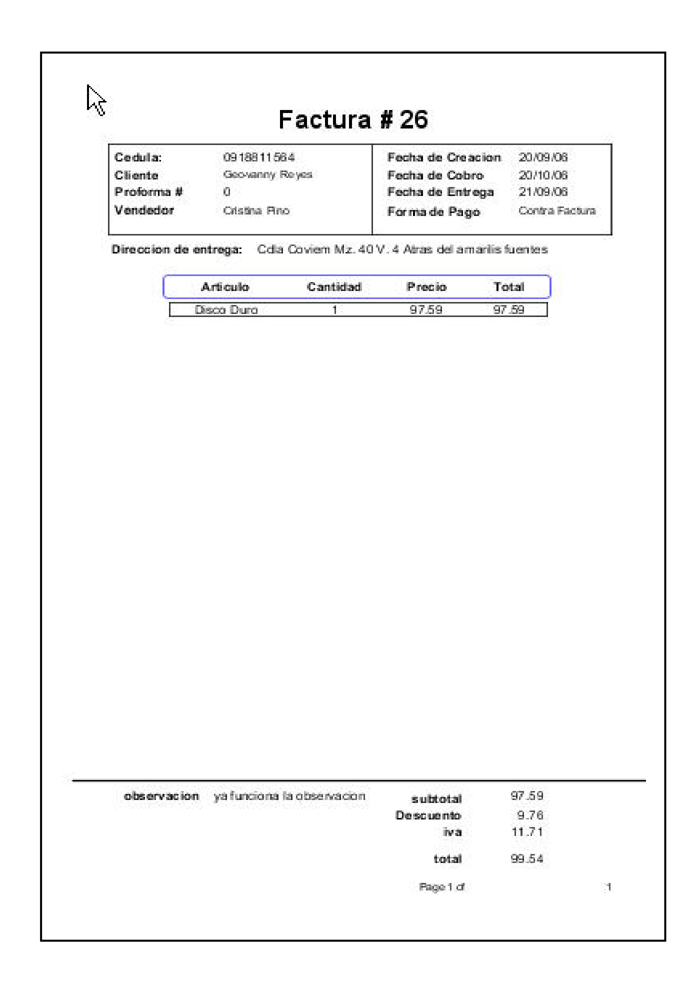
No. Factura	Total Factura	Fecha Emision	Vendedor
31	99.54	2006-09-20	Cristina Pino
30	99.54	2006-09-20	Cristina Pino
29	99.54	2006-09-20	Cristina Pino
28	99.54	2006-09-20	Cristina Pino
27	99.54	2006-09-20	Cristina Pino
26	99.54	2006-09-20	Cristina Pino
25	20.03	2006-09-20	Gabriel Gallo
24	81.32	2006-09-20	Gabriel Gallo
23	24.76	2006-09-20	Gabriel Gallo
22	12.20	2006-09-20	Cristina Pino
21	40.00	2006-09-19	Cristina Pino
20	46.78	2006-09-19	Cristina Pino
19	99.54	2006-09-19	Gabriel Gallo

Por vendedor: El usuario podrá realizar una consulta de todas las facturas que ha realizado un vendedor específico.



No. Factura	Total Factura	Fecha Emision	Vendedor
25	20.03	2006-09-20	Gabriel Gallo
24	81.32	2006-09-20	Gabriel Gallo
23	24.76	2006-09-20	Gabriel Gallo
19	99.54	2006-09-19	Gabriel Gallo
18	79.74	2006-09-19	Gabriel Gallo
17	79.74	2006-09-19	Gabriel Gallo
7	131.95	2006-09-14	Gabriel Gallo

Si se desea ver la descripción completa de toda la factura se puede hacer clic sobre el número de la factura en cualquiera de los criterios de consulta.



Factura # 26

Cedula:	0918811584	**Fecha de Creacion**	20/09/06
Cliente	Geovanny Reyes	**Fecha de Cobro**	20/10/06
Proforma #	0	**Fecha de Entrega**	21/09/06
Vendedor	Cristina Pino	**Forma de Pago**	Contra Factura

Direccion de entrega: Cdla Coviem Mz. 40 V. 4 Atras del amarilis fuentes

Articulo	Cantidad	Precio	Total
Disco Duro	1	97.59	97.59

observacion ya funciona la observacion

subtotal	97.59
Descuento	9.76
iva	11.71
total	99.54

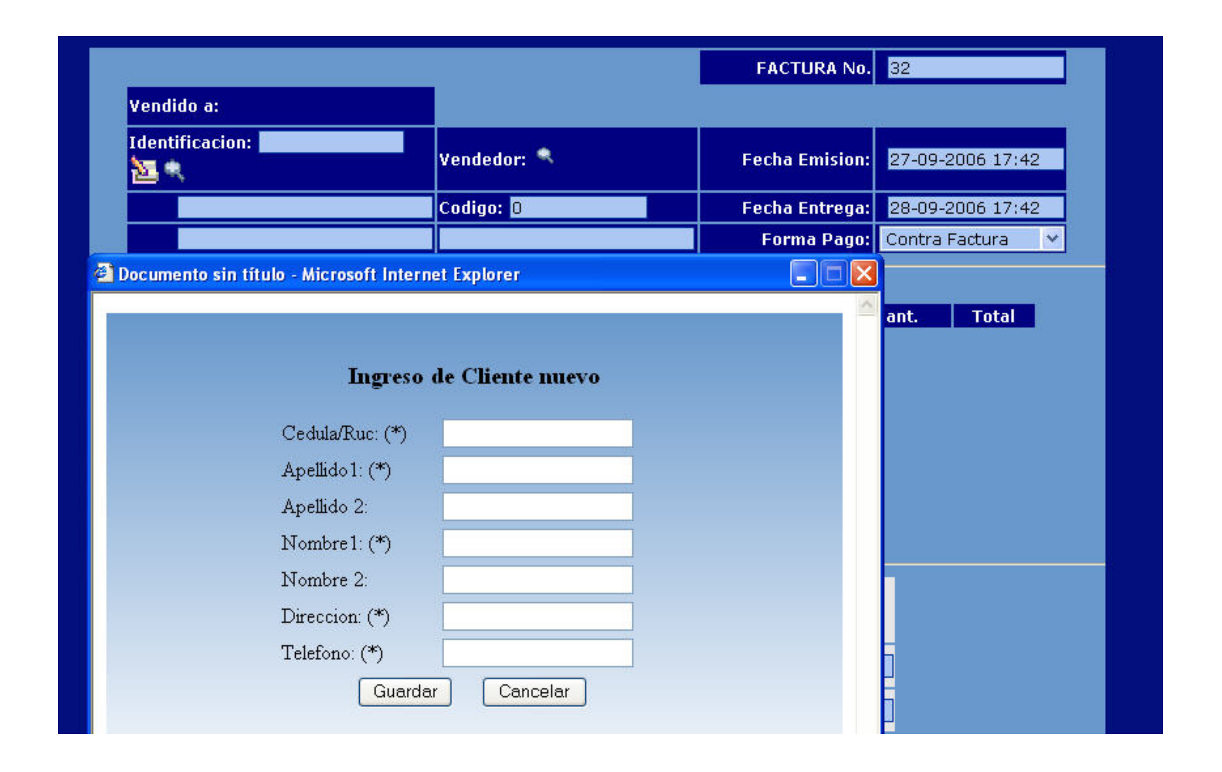
Page 1 of 1

Creación de Facturas

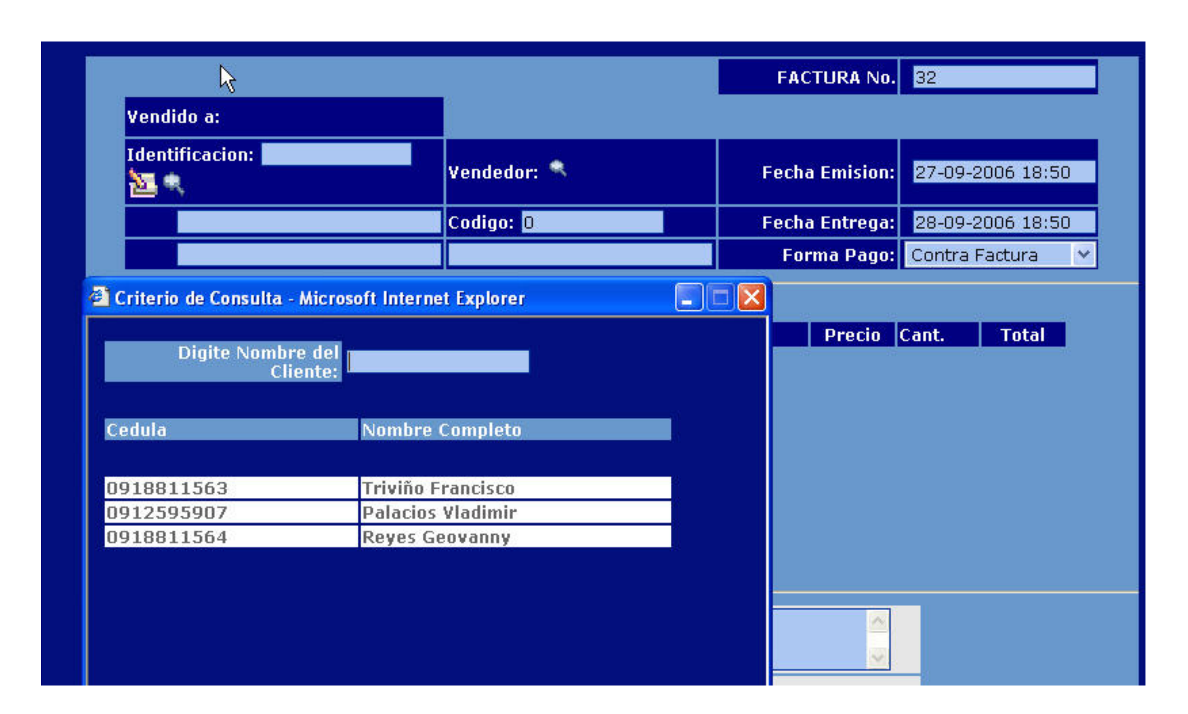
Accediendo a la opción crear factura la aplicación muestra el número correspondiente de la nueva factura a ingresar.

En su cabecera debe ingresar la siguiente información:

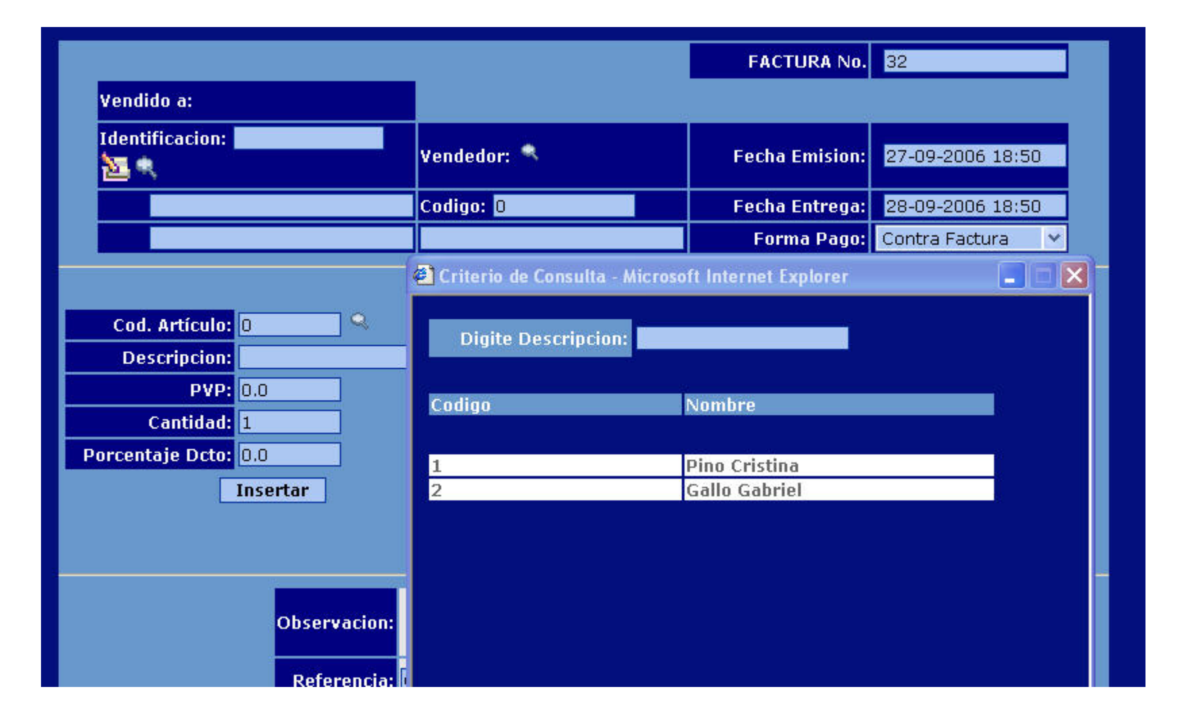
Haciendo clic en el icono del lápiz puede registrar un nuevo cliente donde debe ingresar los siguientes campos: Cédula, Apellido, Nombre, Dirección y Teléfono.



Si es un cliente ya registrado deberá presionar sobre la lupa para seleccionarlo de una lista.



El usuario que realice la venta debe seleccionar su nombre de las lista presionando sobre la lupa.



La fecha de emisión corresponderá a la fecha actual y la fecha de entrega un día después.

La forma de pago que lo determinará el cliente podrá ser de a contado o crédito.

En el contenido de la factura puede ingresar el código del producto si ya lo conoce, caso contrario, puede presionar sobre la lupa, donde le aparece una ventana con las siguientes alternativas:

Criterio de Consulta - Microsoft Internet Explorer

Digite Descripcion:

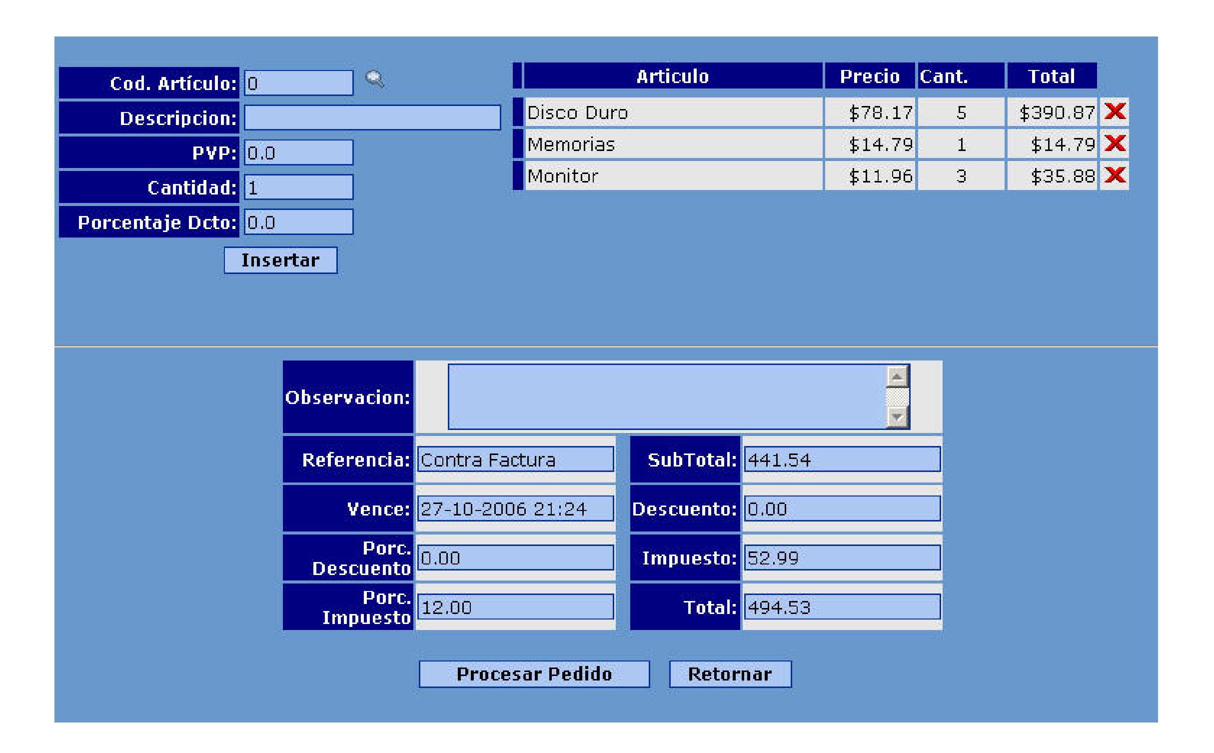
Codigo	Descripción	Precio:	Stock
1	Disco Duro	80.52	132
2	Case	30.00	29
3	Mouse	12.00	23
4	Teclado	50.45	54
5	Monitor	12.32	47
6	Memorias	15.23	61
7	Mainboard	20.45	96
8	Disipador	30.23	56
9	Procesador	100.25	84
10	Transistor	50.56	6

Si la lista de artículos resulta demasiado grande, puede digitar la descripción del producto que esté buscando y en base a esto le mostrará sólo los artículos que guarden esta relación.

Si el producto que busca es visible simplemente selecciónelo que se cargará toda la información del producto en el contenedor, es decir: el código, la descripción, el precio de venta y el porcentaje de descuento.

A partir de este momento puede ingresar la cantidad, que está seteada con valor de uno. Si el vendedor gana por diferencia tiene la opción de cambiar el precio de venta por un precio mayor para generar su comisión (Ver más adelante comisiones).

Una vez que presione el botón insertar se cargará el precio, la cantidad y el total de ese detalle de factura, de forma automática calcula el subtotal, el descuento, el impuesto y el total de la factura, todo esto, de forma sucesiva cada vez que se ingresa un nuevo detalle.



El usuario dispone de la opción de eliminar un detalle si es que lo prefiere el cliente, simplemente presionando sobre la X que se encuentra junto al detalle.

Ya en la parte inferior de la factura puede ingresar una observación si lo desea, además visualizar la fecha de vencimiento que por defecto es un mes, ver el porcentaje de descuento y el porcentaje de impuesto.

Una vez elaborada la factura presione el botón de procesar pedido para guardar la factura e imprimirla o si no lo desea retornar al menú principal.

NOTA DE VENTA

Las opciones en las notas de venta son similares a la factura por lo que se omiten las imágenes.

Consulta de Notas de venta

Accediendo a esta sección del programa puede visualizar de forma directa las notas de venta que han sido elaboradas durante el mes corriente.

En la parte superior puede seleccionar el criterio de consulta más conveniente de acuerdo a sus necesidades:

Por número de nota de venta: Cuando selecciona esta opción se habilita un campo donde puede ingresar el número correspondiente a la nota de venta que desea consultar.

Por cliente. Al seleccionar esta opción, puede consultar de acuerdo al nombre del cliente al que se le hizo una venta.

Por monto: Aquí puede el usuario ingresar un monto inicial y un monto final para realizar la consulta de todas las notas de venta cuyos totales estén comprendidos en este rango.

Por fechas: El usuario podrá ingresar una fecha inicial y una fecha final para consultar todas las notas de venta que se hayan realizado en un período determinado. Puedo optar por seleccionar la fecha deseada directamente del calendario presionando en los botones que se encuentran junto a la caja de texto.

Por vendedor: El usuario podrá realizar una consulta de todas las notas de venta que ha realizado un vendedor específico.

Si se desea ver la descripción completa de toda la nota de venta se puede hacer clic sobre el número de la notas de venta en cualquiera de los criterios de consulta.

Creación de Notas de Venta

Accediendo a la opción crear nota de venta la aplicación muestra el número correspondiente de la nueva nota de venta a ingresar.

En su cabecera debe ingresar la siguiente información:

Haciendo clic en el icono del lápiz puede registrar un nuevo cliente donde debe ingresar los siguientes campos: Cédula, Apellido, Nombre, Dirección y Teléfono.

Si es un cliente ya registrado deberá presionar sobre la lupa para seleccionarlo de una lista.

El usuario que realice la venta debe seleccionar su nombre de las lista presionando sobre la lupa.

La fecha de emisión corresponderá a la fecha actual y la fecha de entrega un día después.

La forma de pago que lo determinará el cliente podrá ser de a contado o crédito.

En el contenido de la nota de venta puede ingresar el código del producto si ya lo conoce, caso contrario, puede presionar sobre la lupa, donde le aparece una ventana con las siguientes alternativas:

Si la lista de artículos resulta demasiado grande, puede digitar la descripción del producto que esté buscando y en base a esto le mostrará sólo los artículos que guarden esta relación.

Si el producto que busca es visible simplemente selecciónelo que se cargará toda la información del producto en el contenedor, es decir: el código, la descripción, el precio de venta y el porcentaje de descuento.

A partir de este momento puede ingresar la cantidad, que está seteada con valor de uno. Si el vendedor gana por diferencia tiene la opción de cambiar el precio de venta por un precio mayor para generar su comisión (Ver más adelante comisiones).

Una vez que presione el botón insertar se cargará el precio, la cantidad y el total de ese detalle de la nota de venta, de forma automática calcula el subtotal, el descuento y el total de la nota de venta, todo esto, de forma sucesiva cada vez que se ingresa un nuevo detalle.

El usuario dispone de la opción de eliminar un detalle si es que lo prefiere el cliente, simplemente presionando sobre la X que se encuentra junto al detalle.

Ya en la parte inferior de la nota de venta puede ingresar una observación si lo desea, además visualizar la fecha de vencimiento que por defecto es un mes y ver el porcentaje de descuento.

Una vez elaborada la nota de venta presione el botón de procesar pedido para guardar la nota de venta e imprimirla o si no lo desea retornar al menú principal.

PROFORMA

Las opciones en la proforma son similares a la factura por lo que se omiten las imágenes.

Consulta de Proformas

Accediendo a esta sección del programa puede visualizar de forma directa las proformas que han sido elaboradas durante el mes corriente.

En la parte superior puede seleccionar el criterio de consulta más conveniente de acuerdo a sus necesidades:

Por número de proforma. Cuando selecciona esta opción se habilita un campo donde puede ingresar el número correspondiente a la proforma que desea consultar.

Por cliente. Al seleccionar esta opción, puede consultar de acuerdo al nombre del cliente al que se le hizo una cotización.

Por monto: Aquí puede el usuario ingresar un monto inicial y un monto final para realizar la consulta de todas las proformas cuyos totales estén comprendidos en este rango.

Por fechas: El usuario podrá ingresar una fecha inicial y una fecha final para consultar todas las proformas que se hayan realizado en un período determinado. Puedo optar por seleccionar la fecha deseada directamente del calendario presionando en los botones que se encuentran a lado de la caja de texto.

Por vendedor: El usuario podrá realizar una consulta de todas las proformas que ha realizado un vendedor específico.

Si se desea ver la descripción completa de toda la proforma se puede hacer clic sobre el número de la proforma en cualquiera de los criterios de consulta.

Creación de Proformas

Accediendo a la opción crear proforma la aplicación muestra el número correspondiente de la nueva proforma a ingresar.

En su cabecera debe ingresar la siguiente información:

Haciendo clic en el icono del lápiz puede registrar un nuevo cliente donde debe ingresar los siguientes campos: Cédula, Apellido, Nombre, Dirección y Teléfono.

Si es un cliente ya registrado deberá presionar sobre la lupa para seleccionarlo de una lista.

El usuario que realice la venta debe seleccionar su nombre de las lista presionando sobre la lupa.

La fecha de emisión corresponderá a la fecha actual y la fecha de entrega un día después.

La forma de pago que lo determinará el cliente podrá ser de a contado o crédito.

En el contenido de la proforma puede ingresar el código del producto si ya lo conoce, caso contrario, puede presionar sobre la lupa, donde le aparece una ventana con las siguientes alternativas:

Si la lista de artículos resulta demasiado grande, puede digitar la descripción del producto que esté buscando y en base a esto le mostrará sólo los artículos que guarden esta relación.

Si el producto que busca es visible simplemente selecciónelo que se cargará toda la información del producto en el contenedor, es decir: el código, la descripción, el precio de venta y el porcentaje de descuento.

A partir de este momento puede ingresar la cantidad, que está seteada con valor de uno. Si el vendedor gana por diferencia tiene la opción de cambiar el precio de venta por un precio mayor para generar su comisión (Ver más adelante comisiones).

Una vez que presione el botón insertar se cargará el precio, la cantidad y el total de ese detalle de proforma, de forma automática calcula el subtotal, el descuento, el impuesto y el total de la proforma, todo esto, de forma sucesiva cada vez que se ingresa un nuevo detalle.

El usuario dispone de la opción de eliminar un detalle si es que lo prefiere el cliente, simplemente presionando sobre la X que se encuentra junto al detalle.

Ya en la parte inferior de la proforma puede ingresar una observación si lo desea, además visualizar la fecha de vencimiento que por defecto es un mes, ver el porcentaje de descuento y el porcentaje de impuesto.

Una vez elaborada la proforma presione el botón de procesar pedido para guardar la proforma e imprimirla o si no lo desea retornar al menú principal.

NOTA DE CRÉDITO

Las opciones en las notas de crédito son similares a la factura por lo que se omiten las imágenes.

Consulta de Nota de Crédito

Accediendo a esta sección del programa puede visualizar de forma directa las notas de crédito que han sido elaboradas durante el mes corriente.

En la parte superior puede seleccionar el criterio de consulta más conveniente de acuerdo a sus necesidades:

Por número de notas de crédito. Cuando selecciona esta opción se habilita un campo donde puede ingresar el número correspondiente a la nota de crédito que desea consultar.

Por cliente. Al seleccionar esta opción, puede consultar de acuerdo al nombre del cliente al que se le hizo una devolución.

Por monto: Aquí puede el usuario ingresar un monto inicial y un monto final para realizar la consulta de todas las notas de crédito cuyos totales estén comprendidos en este rango.

Por fechas: El usuario podrá ingresar una fecha inicial y una fecha final para consultar todas las notas de crédito que se hayan realizado en un período determinado. Puedo optar por seleccionar la fecha deseada directamente del calendario presionando en los botones que se encuentran a lado de la caja de texto.

Por vendedor: El usuario podrá realizar una consulta de todas las notas de crédito que ha realizado un vendedor específico.

Si se desea ver la descripción completa de toda la nota de crédito se puede hacer clic sobre el número de la nota de crédito en cualquiera de los criterios de consulta.

Creación de Nota de Crédito

Accediendo a la opción crear nota de crédito la aplicación muestra el número correspondiente de la nueva nota de crédito a ingresar.

En su cabecera debe ingresar la siguiente información:

Seleccionar el nombre del cliente presionando sobre la lupa.

El usuario que realice la devolución debe seleccionar su nombre de las lista presionando sobre la lupa.

La fecha de emisión corresponderá a la fecha actual y la fecha de entrega un día después.

En el contenido de la nota de crédito puede ingresar el código del producto si ya lo conoce, caso contrario, puede presionar sobre la lupa, donde le aparece una ventana con las siguientes alternativas:

Si la lista de artículos resulta demasiado grande, puede digitar la descripción del producto que esté buscando y en base a esto le mostrará sólo los artículos que guarden esta relación.

Si el producto que busca es visible simplemente selecciónelo que se cargará toda la información del producto en el contenedor, es decir: el código, la descripción, el precio de venta y el porcentaje de descuento.
A partir de este momento puede ingresar la cantidad, que está seteada con valor de uno.

Una vez que presione el botón insertar se cargará el precio, la cantidad y el total de ese detalle de factura, de forma automática calcula el subtotal, el descuento, el impuesto y el total de la nota de crédito, todo esto, de forma sucesiva cada vez que se ingresa un nuevo detalle.

El usuario dispone de la opción de eliminar un detalle si es que existe una equivocación, simplemente presionando sobre la X que se encuentra junto al detalle.

Ya en la parte inferior de la nota de crédito puede ingresar una observación si lo desea, además visualizar la fecha de vencimiento que por defecto es un mes, ver el porcentaje de descuento y el porcentaje de impuesto.

Una vez elaborada la nota de crédito presione el botón de procesar pedido para guardar e imprimirla o si no lo desea retornar al menú principal.

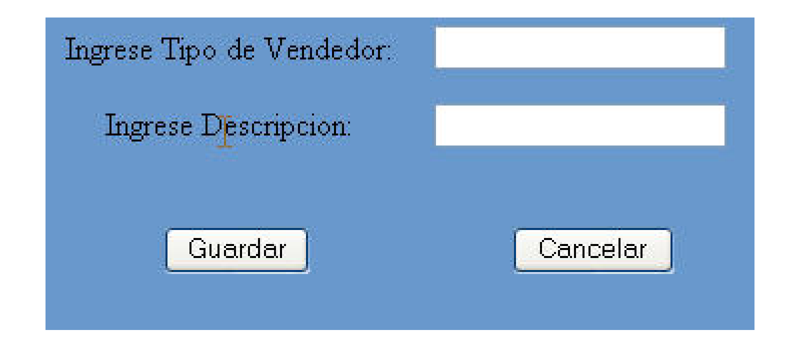
VENDEDORES

Al acceder a esta sección del programa dispone de tres opciones:

Crear Nuevo Vendedor: Puede ingresar en el sistema un nuevo vendedor, ingresando el código de un empleado que ya esté registrado en la empresa. A continuación deberá ingresar el nombre y el apellido del empleado y seleccionar el tipo de vendedor y el tipo de comisión a ganar.

Presione el botón guardar para registrar el nuevo vendedor o simplemente cancele.

Crear Nuevo Tipo de Vendedor: Si la empresa así lo decide y de acuerdo a las políticas en el área de ventas se puede crear una nueva de clase de vendedor. Deberá ingresar el nuevo tipo y la descripción.



Crear un Nuevo Tipo de Comisión: Si la empresa así lo decide y de acuerdo a las políticas en el área de ventas se puede crear una nueva de clase de comisión. Deberá ingresar el nuevo tipo y la descripción.



COMISIONES

En esta sección del programa podrá calcular de forma directa la comisión del mes de todos los vendedores sin importar el tipo.



Codigo Vendedor	Nombre	Tipo Comision	Mes Comision	No. Ventas	Valor Venta	Valor Devolucion	Comision
1	Cristina Pino	Comision por Diferencia	02	26	2822.79	161.13	**33.29**
2	Gabriel Gallo	Comision por Porcentaje	02	8	517.08	0.00	**77.56**

Para que el vendedor pueda ganar comisiones deberá cumplir con las siguientes reglas:

Que las facturas generadas tengan el estado P de pagadas.

Que los montos de ventas estén comprendidos entre los siguientes valores para ganar los siguientes porcentajes:

500 – 1000, el 2%

1001 – 2000, el 3%

2001 – 3000, el 5%

3000 – 100000, el 10%

Como se realiza un seguimiento del número de ventas de los vendedores, para que en un sola venta o en pocas puedan alcanzar los montos deseados: absolutamente (de ese porcentaje ya ganado por ellos) entra a penalizarse

el 50%, con los siguientes parámetros de acuerdo a las número de ventas conseguidos en el mes:

1 – 3, el 90% de castigo.

4 – 10, el 50% de castigo.

11 – 20, el 30% de castigo.

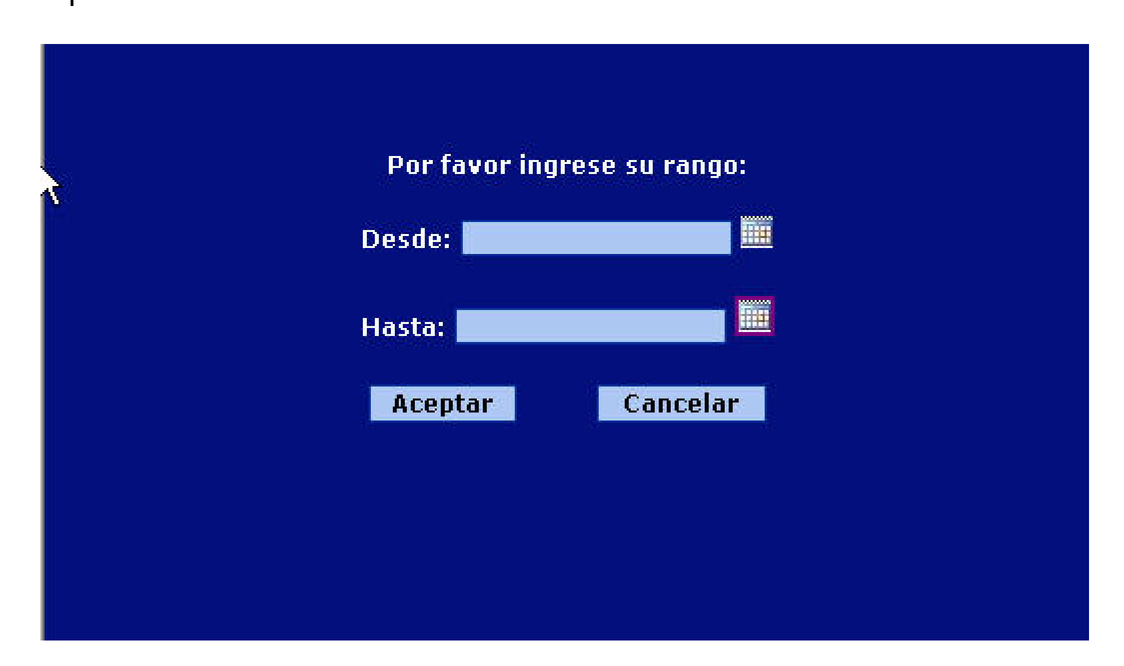
21 – 30, el 10% de castigo.

31 – 1000, 0 % de castigo.

REPORTES

En esta parte del sistema puede crear los siguientes tipos de reportes:

Ventas por Fecha. En este tipo de reporte debe seleccionar una fecha inicial y una fecha final, es decir un período de tiempo para realizar el reporte.



Ventas por Fecha

jueves 28 septiembre 2006

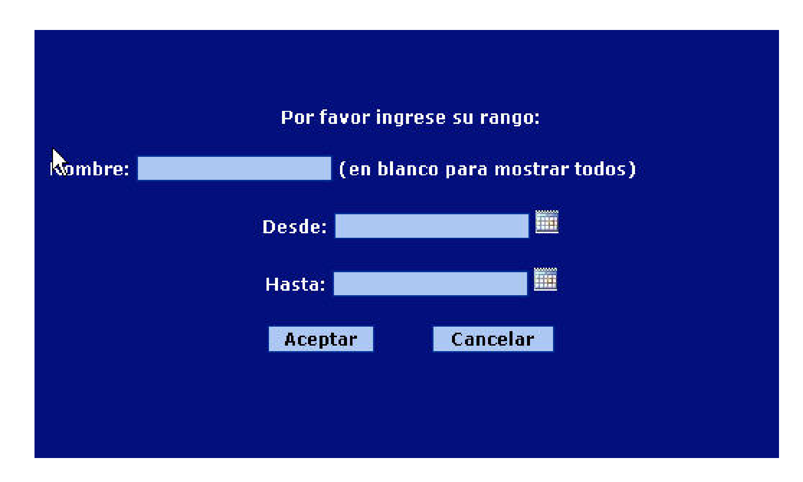
Fecha	Cliente	No. Doc	Tipo Doc	Items x Fac.	Subtotal	Desc	Imp.	Total
21-sep-06	Triviño Francisco Xavier	3	N/E	1	97.59	9.76	0.0	87.83
20-sep-06	Reyes Geovanny	26	FAC	1	97.59	9.76	11.71	99.54
20-sep-06	Palacios Vladimir	22	FAC	1	12.0	1.2	1.44	12.2
20-sep-06	Reyes Geovanny	31	FAC	1	97.59	9.76	11.71	99.54
20-sep-06	Triviño Francisco Xavier	24	FAC	1	79.73	7.97	9.57	81.32
20-sep-06	Triviño Francisco Xavier	23	FAC	1	24.27	2.43	2.91	24.76
20-sep-06	Reyes Geovanny	28	FAC	1	79.0	9.76	11.71	99.54
20-sep-06	Triviño Francisco Xavier	25	FAC	1	16.0	1.96	2.36	20.03
20-sep-06	Reyes Geovanny	27	FAC	1	78.17	9.76	11.71	99.54
20-sep-06	Reyes Geovanny	29	FAC	1	79.0	9.76	11.71	99.54
20-sep-06	Reyes Geovanny	30	FAC	1	79.0	9.76	11.71	99.54
19-sep-06	Reyes Geovanny	14	FAC	1	11.96	1.2	1.44	12.2
19-sep-06	Palacios Vladimir	21	FAC	1	39.22	3.92	4.71	40.0
19-sep-06	Reyes Geovanny	17	FAC	1	78.17	7.82	9.38	79.74
19-sep-06	Reyes Geovanny	16	FAC	1	35.88	3.59	4.31	36.6
19-sep-06	Palacios Vladimir	18	FAC	1	97.59	7.82	9.38	79.74
19-sep-06	Reyes Geovanny	20	FAC	1	50.0	4.59	5.5	46.78
19-sep-06	Reyes Geovanny	19	FAC	1	80.0	9.76	11.71	99.54
19-sep-06	Reyes Geovanny	15	FAC	1	11.96	1.2	1.44	12.2
19-sep-06	Reyes Geovanny	13	FAC	1	11.96	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	9	FAC	1	11.96	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	12	FAC	1	11.96	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	8	FAC	1	11.96	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	11	FAC	1	11.96	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	10	FAC	1	11.96	1.2	1.44	12.2
15-sep-06	Triviño Francisco Xavier	2	N/E	1	11.96	1.2	0.0	10.77
14-sep-06	Palacios Vladimir	3	FAC	1	20.0	1.2	1.44	12.2
14-sep-06	Triviño Francisco Xavier	2	FAC	1	11.96	1.2	1.44	12.2
14-sep-06	Triviño Francisco Xavier	5	FAC	1	1090.91	109.09	130.91	1112.73
14-sep-06	Palacios Vladimir	7	FAC	2	129.36	12.94	15.52	131.95
14-sep-06	Reyes Geovanny	6	FAC	1	11.96	1.2	1.44	12.2
14-sep-06	Reyes Geovanny	1	N/E	1	45.86	4.59	0.0	41.28
14-sep-06	Reyes Geovanny	4	FAC	1	30.0	1.2	1.44	12.2
14-sep-06	Palacios Vladimir	1	FAC	2	872.2	87.22	104.66	889.64

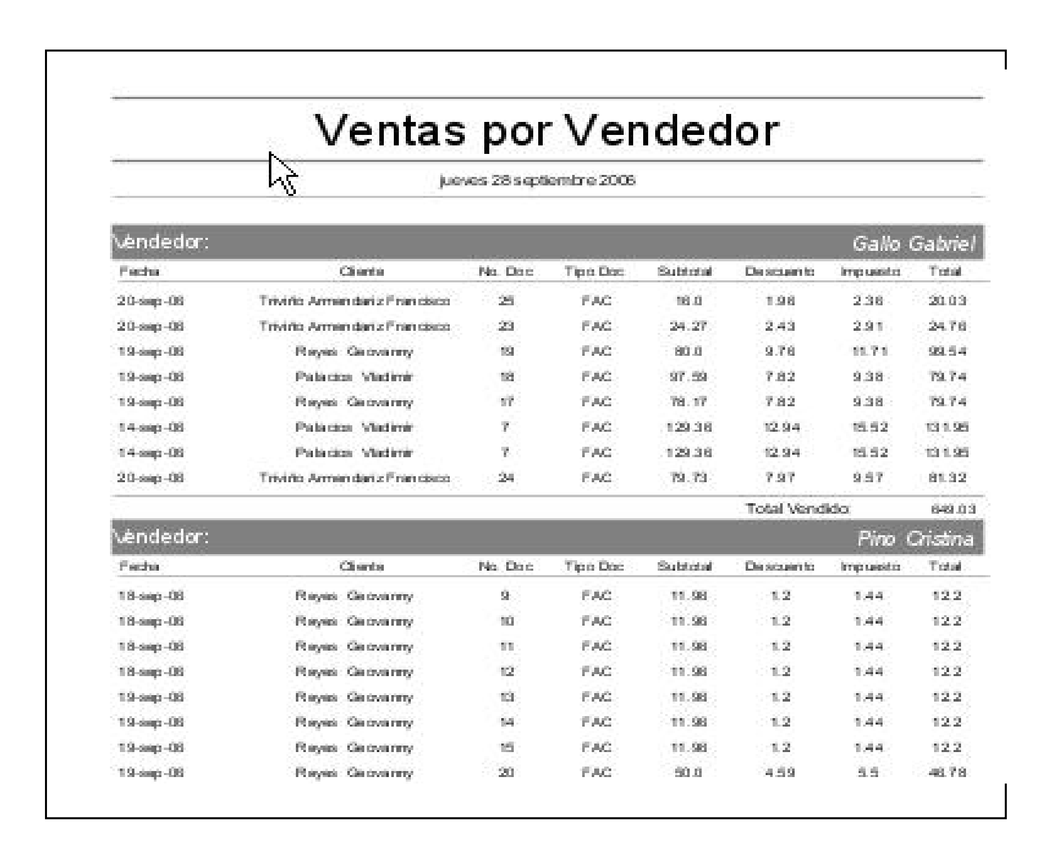
Page 1 of 1

Total Vendido: 3538.55

En el reporte se señala la fecha de cada venta junto con toda la información de correspondiente a cada factura. En la parte señal indica el total vendido en el período ingresado.

Ventas por Vendedor. En este tipo de reporte debe ingresar el nombre del vendedor, junto con un fecha incio y fin. Si no digita ningún nombre muestra de forma automática el reporte de todos los vendedores en ese período de tiempo.





Ventas por Vendedor

jueves 28 septiembre 2006

Vendedor: *Gallo Gabriel*

Fecha	Cliente	No. Doc	Tipo Doc	Subtotal	Descuento	Impuesto	Total
20-sep-06	Triviño Armendariz Francisco	25	FAC	18.0	1.98	2.38	20.03
20-sep-06	Triviño Armendariz Francisco	23	FAC	24.27	2.43	2.91	24.78
19-sep-06	Reyes Geovanny	19	FAC	80.0	9.78	11.71	98.54
19-sep-06	Palacios Vladimir	18	FAC	97.59	7.82	9.38	79.74
19-sep-06	Reyes Geovanny	17	FAC	78.17	7.82	9.38	79.74
14-sep-06	Palacios Vladimir	7	FAC	129.38	12.94	15.52	131.96
14-sep-06	Palacios Vladimir	7	FAC	129.38	12.94	15.52	131.96
20-sep-06	Triviño Armendariz Francisco	24	FAC	79.73	7.97	9.57	81.32
					Total Vendido:		649.03

Vendedor: *Pino Cristina*

Fecha	Cliente	No. Doc	Tipo Doc	Subtotal	Descuento	Impuesto	Total
18-sep-06	Reyes Geovanny	9	FAC	11.98	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	10	FAC	11.98	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	11	FAC	11.98	1.2	1.44	12.2
18-sep-06	Reyes Geovanny	12	FAC	11.98	1.2	1.44	12.2
19-sep-06	Reyes Geovanny	13	FAC	11.98	1.2	1.44	12.2
19-sep-06	Reyes Geovanny	14	FAC	11.98	1.2	1.44	12.2
19-sep-06	Reyes Geovanny	15	FAC	11.98	1.2	1.44	12.2
19-sep-06	Reyes Geovanny	20	FAC	50.0	4.59	5.5	48.78

Devoluciones por fecha. Permite hacer un reporte de las devoluciones que se han realizado en un período determinado. De igual forma forma debe ingresar una fecha inicial yotra final.

Devoluciones por Fecha

No. Nota de credito: *4*

Cliente: *Palacios Vladimir*

Fecha	Codigo	Articulo	Cantidad	Precio	Subtotal
20-sep-06	1	Disco Duro	1	97.59	97.59
			Total:		97.59

No. Nota de credito: *3*

Cliente: *Palacios Vladimir*

Fecha	Codigo	Articulo	Cantidad	Precio	Subtotal
20-sep-06	1	Disco Duro	1	97.59	97.59
			Total:		97.59

No. Nota de credito: *2*

Cliente: *Reyes Geovanny*

Fecha	Codigo	Articulo	Cantidad	Precio	Subtotal
15-sep-06	6	Memorias	1	14.79	14.79
			Total:		14.79

Ventas de Artículos. Permite realizar reportes por artículos. Es decir detalla cuantas unidades fueron vendidas de un procucto, cuanto fueron sus totales y porcentajes de venta.

El reporte detalla tanto las facturas como las notas de venta.

Venta Articulos Consolidado

Tipo Doc. FAC

Codigo	Producto	Precio normal	Precio Regular	U. Vendidas	Precio Promedio	V. Venta total	Diferencia	%
1	Disco Duro	80.52	78.17	10	96.61	966.19	184.49	12.20 %
2	Case	30.00	29.41	21	39.22	823.53	205.92	25.61 %
4	Teclado	50.45	45.86	21	50.20	1136.77	173.71	25.61 %
5	Monitor	12.32	11.96	17	11.96	203.32	0.00	20.73 %
6	Memorias	15.23	14.78	6	21.28	123.64	34.96	7.32 %
8	Disipador	30.23	29.34	4	29.35	117.40	0.04	4.88 %

Tipo Doc. NVE

Codigo	Producto	Precio normal	Precio Regular	U. Vendidas	Precio Promedio	V. Venta total	Diferencia	%
1	Disco Duro	80.52	78.17	1	97.59	97.59	19.42	1.22 %
4	Teclado	50.45	45.86	1	45.86	45.86	0.00	1.22 %
5	Monitor	12.32	11.96	1	11.96	11.96	0.00	1.22 %

Libro de Ventas Realiza el reporte de todas las ventas hechas a través de facturas o notas de ventas en un período de tiempo determinado. Disminuye el total de acuerdo a las devoluciones hechas en ese lapso.

Libro de Ventas

jueves 28 septiembre 2006

Fecha	Cliente	No. Doc	Tipo Doc	Subtotal	Desc	Imp.	Total
15/09/06 12:00	Reyes Geovanny	2	NCR	14.79	1.48	1.77	-15.08
15/09/06 12:00	Triviño Francisco	2	NVE	11.96	1.2	0.0	10.77
14/09/06 12:00	Reyes Geovanny	4	FAC	30.0	1.2	1.44	12.2
14/09/06 12:00	Palacios Vladimir	1	FAC	872.2	87.22	104.66	889.64
14/09/06 12:00	Reyes Geovanny	6	FAC	11.96	1.2	1.44	12.2
14/09/06 12:00	Palacios Vladimir	7	FAC	129.36	12.94	15.52	131.95
14/09/06 12:00	Reyes Geovanny	1	NVE	45.86	4.59	0.0	41.28
14/09/06 12:00	Triviño Francisco	5	FAC	1090.91	109.09	130.91	1112.73
14/09/06 12:00	Palacios Vladimir	3	FAC	20.0	1.2	1.44	12.2
14/09/06 12:00	Triviño Francisco	2	FAC	11.96	1.2	1.44	12.2

Ventas por Clientes. Realiza el reporte de ventas de un cliente determinado. Si no digita el cliente muestra todos los clientes que han realizado compras en la fecha señalada.

Ventas por Cliente

jueves 28 septiembre 2006

Cliente: *Palacios Vladimir*

Fecha	No. Doc	Tipo Doc	Items en Doc.	Subtotal	Descuento	Impuesto	Total
14/09/06 12:00	3	FAC	1	20.0	1.2	1.44	12.2
14/09/06 12:00	1	FAC	2	872.2	87.22	104.66	889.64
19/09/06 12:00	18	FAC	1	97.59	7.82	9.38	79.74
14/09/06 12:00	7	FAC	2	129.36	12.94	15.52	131.95
20/09/06 12:00	22	FAC	1	12.0	1.2	1.44	12.2
19/09/06 12:00	21	FAC	1	39.22	3.92	4.71	40.0

Total Adquirido: 1165.73

Cliente: *Reyes Geovanny*

Fecha	No. Doc	Tipo Doc	Items en Doc.	Subtotal	Descuento	Impuesto	Total
20/09/06 12:00	28	FAC	1	79.0	9.76	11.71	99.54
20/09/06 12:00	29	FAC	1	79.0	9.76	11.71	99.54
20/09/06 12:00	30	FAC	1	79.0	9.76	11.71	99.54
20/09/06 12:00	31	FAC	1	97.59	9.76	11.71	99.54
18/09/06 12:00	9	FAC	1	11.96	1.2	1.44	12.2
18/09/06 12:00	10	FAC	1	11.96	1.2	1.44	12.2

Artículos más vendidos. Detalla los artículos que más se han vendido en el mes.

Productos mas Vendidos

lunes 09 octubre 2006

Producto	*Disipador*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
29.34	35.27	1285.43	3.0	35.0

Producto	*Mainboard*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
19.85	20.53	270.34	3.0	13.0

Producto	*Disco Duro*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
78.17	78.89	1033.51	2.0	13.0

Producto	*Mouse*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
11.65	13.10	168.93	4.0	12.0

Producto	*Teclado*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
45.86	56.82	490.91	2.0	9.0

Producto	*Procesador*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
98.28	117.65	941.18	1.0	8.0

Producto	*Monitor*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
11.96	13.26	99.34	2.0	7.0

Producto	*Memorias*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
14.78	23.30	163.11	1.0	7.0
14.78	14.79	14.79	1.0	1.0

Producto	*Case*			
Precio Normal	**P. Promedio**	**Venta Acum.**	**Doc. Emitidos**	**U. Vendidas**
29.41	29.41	29.41	1.0	1.0

General Vendedores. Detalla el número de ventas (entre facturas y notas de ventas) y el monto total conseguido por cada vendedor para determinar el rendimiento de cada empleado.



Ventas Totales por Vendedor

lunes 09 octubre 2006

Vendedor: *Gallo Gabriel*

No. Ventas	11.0
Total Vendido	2789.61

Vendedor: *Pino Cristina*

No. Ventas	14.0
Total Vendido	5083.81

Reporte de Comisiones Generales. Detalla de forma específica el valor y el tipo de comisión de cada vendedor.

General de Comisiones

Fecha *octubre-06*

Nombre	Tipo de Comision	Comision Mensual
Gallo Gabriel	Comision por Porcentaje	948.46
Pino Cristina	Comision por Diferencia	699.26

Lista de Precios.- Permite visualizar o imprimir la lista de precios para entregar a los vendedores.

Lista de Precios

lunes 09 octubre 2006

Codigo	Descripcion	Stock	Precio	% Desc.
1	Disco Duro	986	80.52	3.00
2	Case	998	30.00	2.00
3	Mouse	988	12.00	3.00
4	Teclado	998	50.45	10.00
5	Monitor	993	12.32	3.00
6	Memorias	994	15.23	3.00
7	Mainboard	987	20.45	3.00
8	Disipador	965	30.23	3.00
9	Procesador	992	100.25	2.00
10	Transistor	1001	50.56	1.00